ΛΟΓΙΣΜΙΚΑ ΜΟΝΤΕΛΑ ΑΝΤΙΜΕΤΩΠΙΣΗΣ ΠΥΡΚΑΓΙΑΣ

ΞΕΝΟΦΩΝ Ι. ΔΡΑΚΩΤΟΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ, ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ
ΒΟΛΟΣ

ΙΟΥΛΙΟΣ 2007

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Με το παρόν, οι υπογράφοντες πιστοποιούν και βεβαιώνουν το τμήμα ότι ο **ΞΕΝΟΦΩΝ Ι. ΔΡΑΚΩΤΟΣ** με την διπλωματική εργασία με θέμα **"ΛΟΓΙΣΜΙΚΑ ΜΟΝΤΕΛΑ ΑΝΤΙΜΕΤΩΠΙΣΗΣ ΠΥΡΚΑΓΙΑΣ"** ικανοποιεί τις προυποθέσεις για απόκτηση διπλώματος.

Ημερομηνία: ΙΟΥΛΙΟΣ 2007

Επιβλέποντες Καθηγητές: _____
Παναγιώτα Τσομπανοπούλου

_____
Ηλίας Χούστης

Θέλω να ευχαριστήσω την καθηγήτριά μου, κα Παναγιώτα Τσομπανοπούλου για την άψογη συνεργασία, το ενδιαφέρον και το χρόνο που μου διέθεσε για να δημιουργηθεί η παρούσα εργασία.

Επίσης, θέλω να ευχαριστήσω την οικογένειά μου για την πολύτιμη στήριξη που μου προσέφερε όλα αυτά τα χρόνια.

Περιεχόμενα

Περίληψη

Η εργασία είναι εμπνευσμένη από την ανάγκη για ανάπτυξη τεχνολογίας κατάλληλης να παρέχει αξιόπιστη βοήθεια στην λήψη αποφάσεων σε καταστάσεις κρίσης. Ο προσομοιωτής θα μπορεί να προβλέψει την εξέλιξη της φωτιάς για το άμεσο-σύντομο μέλλον, ώστε ο χρήστης να μπορεί να λάβει τη σωστότερη απόφαση για την αντιμετώπιση της φωτιάς στον συγκεκριμένο χώρο. Δίκτυα ασύρματων αισθητήρων χρησιμοποιούνται για την παρακολούθηση του φαινομένου και την μετάδοση των τιμών της θερμοκρασίας στο σύστημα. Προγράμματα προσομοίωσης της εξέλιξης της φωτιάς εκτελούνται στο **Grid** και η τεχνολογία των πρακτόρων χρησιμοποιείται για να συντονίσει το ετερογενές λογισμικό.

Abstract

The paper is motivated by the need to develop technological infrastructure for informed and reliable decision support in crisis situations. The simulator will be capable of predicting fire propagation dynamics to allow the user to take decisions based on reliable microfuture predictions of fire propagation based on potential personnel actions. The **Grid** is utilised for running legacy codes enabling fire propagation prediction. Sensor networks are used to enable simulation steering and correction. Agent **middleware** is utilized to coordinate heterogeneous software.

Εισαγωγή

Η εργασία αυτή έχει παρακινηθεί από την ανάγκη για ανάπτυξη μιας υποδομής για ενημερωμένη και αξιόπιστη υποστήριξη αποφάσεων στις διάφορες καταστάσεις κρίσης και καταστροφής. Στην εργασία αυτή περιγράφεται η υλοποίηση του οδηγούμενου από δεδομένα, προσομοιωτή πυρκαγιάς, ικανού για την πρόβλεψη του τρόπου εξάπλωσης της πυρκαγιάς. Το υλοποιημένο λογισμικό επιτρέπει στο χρήστη να παίρνει αποφάσεις βασισμένες σε αξιόπιστες προβλέψεις του τρόπου εξάπλωσης της πυρκαγιάς και να ερευνά πιθανές ενέργειες του προσωπικού. Μια υπολογιστική υποδομή που αποτελείται από πολλές υπολογιστικές μονάδες (**cluster**) χρησιμοποιείται για το "τρέξιμο" του κώδικα, ο οποίος προβλέπει τον τρόπο εξάπλωσης της πυρκαγιάς. Θα χρησιμοποιηθούν δίκτυα αισθητήρων για να επιτρέψουν την οδήγηση και τη διόρθωση της προσομοίωσης. Επίσης, σταθμοί βάσης χρησιμοποιούνται για το συντονισμό δεδομένων και επικοινωνιών. Οι φορητές συσκευές είναι αρχικά και τερματικά σημεία της αλληλεπίδρασης του προσωπικού κρίσης (δηλ., των πυροσβεστών) με το υπόλοιπο σύστημα. Ακόμα τεχνολογία πρακτόρων αναλαμβάνει να συντονίσει το ετερογενές λογισμικό/υλικό.

Η εργασία αποτελείται από 5 Κεφάλαια. Στο Κεφάλαιο 1 περιγράφονται τα απαραίτητα λογισμικά εργαλεία, ενώ στο Κεφάλαιο 2 παρουσιάζεται ένα απλό πλαίσιο πύλης για δίκτυα αισθητήρων. Η περιγραφή του λογισμικού που δημιουργήσαμε βρίσκεται στο Κεφάλαιο 3, ενώ στο Κεφάλαιο 4 υπάρχει λίστα των πειραμάτων. Τέλος στο Κεφάλαιο 5 περιγράφουμε συμπεράσματα και μελλοντικές επεκτάσεις. Στο Παράρτημα δίνουμε μία περιγραφή για το λογισμικό και τη λειτουργία του.

Κεφάλαιο 1

Απαιτούμενο λογισμικό

1.1 Fire Dynamics Simulation (FDS) - Smokeview

1.1.1 Εισαγωγή

Το **FDS** είναι ένα υπολογιστικό μοντέλο δυναμικής ροών για ροή ρευστών οδηγούμενη από φωτιά. Το λογισμικό είναι ένα υπολογιστικό πρόγραμμα σε **FORTRAN 90**, το οποίο επιλύει αριθμητικά μια μορφή των εξισώσεων Navier-Stokes, κατάλληλες για ροή χαμηλής ταχύτητας, οδηγούμενης από θερμότητα, δίνοντας έμφαση στον καπνό και τη μεταφορά θερμότητας από την φωτιά. Η πρώτη έκδοση του **FDS** δημοσιοποιήθηκε τον Φεβρουάριο του 2000, ενώ η πιο πρόσφατη έκδοση είναι η τέταρτη και δημοσιοποιήθηκε τον Μάρτιο του 2006. Οι διαφορές μεταξύ των εκδόσεων, επικεντρώνονται σε σημαντικές αλλαγές του φυσικού μοντέλου ή διαφορετικές παραμέτρους. Τόσο η ανάπτυξη όσο και η συντήρηση του προσομοιωτή (**Fire Dynamics Simulator**) έχουν πραγματοποιηθεί μέσω μιας συνεργασίας από δημόσιες και ιδιωτικές οργανώσεις, στις Ηνωμένες Πολιτείες και στο εξωτερικό.

Το **Smokeview** είναι ένα πρόγραμμα απεικόνισης γραμμένο σε γλώσσα προγραμματισμού **C/OpenGL** που χρησιμοποιείται για να επιδείξει τα αποτελέσματα μιας προσομοίωσης του **FDS**.

1.1.2 Περιγραφή FDS

Το FDS είναι ένα υπολογιστικό μοντέλο δυναμικής ρευστών (**Computational Fluid Dynamics - CFD**), οδηγούμενης από πυρκαγιά. Το μοντέλο επιλύει αριθμητικά μια μορφή των εξισώσεων Navier-Stokes, κατάλληλων για την χαμηλής ταχύτητας, θερμικά-οδηγούμενης

ροής με έμφαση στον καπνό και τη μεταφορά θερμότητας από τις πυρκαγιές. Οι μερικές παράγωγοι στις εξισώσεις διατήρησης της μάζας, της ορμής και της ενέργειας προσεγγίζονται με πεπερασμένες διαφορές (**finite differences**), και η λύση παράγεται σε κάθε χρονική στιγμή στα σημεία ενός τρισδιάστατου, κυβικού πλέγματος. Ακόμα υπολογίζεται η θερμική ακτινοβολία χρησιμοποιώντας την τεχνική των πεπερασμένων όγκων (**finite volumes**) επάνω στο ίδιο πλέγμα. Σωματίδια **Lagrange** χρησιμοποιούνται για την προσομοίωση της κίνησης του καπνού.

Η ανάπτυξη του **FDS** στοχεύει στην επίλυση των πρακτικών προβλημάτων πυρκαγιάς στην εφαρμοσμένη μηχανική πυροπροστασίας και συγχρόνως στην παροχή ενός εργαλείου για την έρευνα των θεμελιωδών αρχών της φωτιάς και της μελέτης της καύσης. Το **FDS** μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση των ακόλουθων φαινομένων:

- Χαμηλής ταχύτητας μεταφορά θερμότητας και των προϊόντων καύσης από την πυρκαγιά.
- Μεταφορά θερμότητας με ακτινοβολία και μεταγωγή, μεταξύ του αερίου και των στερεών επιφανειών.
- Πυρόλυση.
- Διάδοση της φλόγας και εξέλιξη πυρκαγιάς.
- Ενεργοποίηση ψεκαστήρων, ανιχνευτών θερμότητας και ανιχνευτών καπνού.
- Ψεκασμοί ψεκαστήρων και καταστολή από το νερό.

Το **FDS** υπολογίζει τη θερμοκρασία, την πυκνότητα, την πίεση, την ταχύτητα και τη χημική σύσταση σε κάθε αριθμημένο κελί του πλέγματος σε κάθε διακριτό χρονικό βήμα. Συνήθως, υπάρχουν από εκατοντάδες χιλιάδες έως αρκετά εκατομμύρια κελιά πλέγματος και από χιλιάδες έως εκατοντάδες χιλιάδες χρονικά βήματα. Επιπλέον, το **FDS** υπολογίζει την θερμοκρασία, τη ροή της θερμότητας, το ποσοστό απώλειας μάζας και διάφορες άλλες ποσότητες στις στερεές επιφάνειες. Ο χρήστης πρέπει προσεκτικά να επιλέξει ποια στοιχεία θα αποθηκευτούν, όπως θα έκανε κάποιος στο σχεδιασμό ενός πραγματικού πειράματος. Παρότι μόνο ένα μικρό μέρος των υπολογισμένων πληροφοριών μπορεί να σωθεί, το αποτέλεσμα εξόδου του FDS αποτελείται από μερικά αρκετά μεγάλα αρχεία δεδομένων. Οι τυπικές

χαρακτηριστικές ποσότητες εξόδου, για τη αέρια φάση περιλαμβάνουν:

- Θερμοκρασία αερίων.
- Ταχύτητα αερίων.
- Συγκέντρωση αερίων στοιχείων (υδρατμοί, CO_2, CO, N_2).
- Εκτίμηση συγκέντρωσης και διαφάνειας καπνού.
- Πίεση.
- Ρυθμό απελευθέρωσης θερμότητας ανά μονάδα όγκου.
- Λόγο μιγμάτων (ή αναλογία αέρα/καυσίμου).
- Πυκνότητα αερίου.
- Μάζα σταγονιδίων νερού ανά μονάδα όγκου.

Στις στερεές επιφάνειες, το **FDS** υπολογίζει τις πρόσθετες ποσότητες που συνδέονται με την ισορροπία ενέργειας μεταξύ αέριας και στερεής φάσης, συμπεριλαμβάνοντας:

- Επιφανειακή και εσωτερική θερμοκρασία.
- Ροή θερμότητας από ακτινοβολία και μεταγωγή.
- Ρυθμό καύσης.

- Μάζα υδάτινων σταγονιδίων ανά μονάδα περιοχής.

Οι ποσότητες που καταγράφονται από το πρόγραμμα περιλαμβάνουν:

- Συνολικό ποσοστό απελευθέρωσης θερμότητας (**HRR**).
- Χρόνους ενεργοποίησης ψεκαστήρων και ανιχνευτών.
- Ροές μάζας και ενέργειας μέσω των ανοιγμάτων ή στερεών.

Οι τιμές των διάφορων ποσοτήτων στο χρόνο, όπως το ποσοστό απελευθέρωσης θερμότητας της πυρκαγιάς (**HRR**) σε συγκεκριμένα σημεία ή σε όλα τα σημεία του πλέγματος αποθηκεύονται σε αρχεία κειμένου που μπορούν να σχεδιαστούν χρησιμοποιώντας ένα πρόγραμμα υπολογισμών με λογιστικά φύλλα (**spreadsheet**). Εντούτοις, τα περισσότερα δεδομένα στο χώρο ή στις επιφάνειες του χώρου της προσομοίωσης μπορούν να απεικονιστούν με το πρόγραμμα **Smokeview**, που είναι ειδικά σχεδιασμένο για να αναλύει τα στοιχεία που παράγονται από το **FDS**. Τα **FDS** και **Smokeview** χρησιμοποιούνται μαζί στη διαμόρφωση (μοντελοποίηση) και στην απεικόνιση φαινομένων πυρκαγιάς. Το **Smokeview** απεικονίζει την πυρκαγιά παρουσιάζοντας την κίνηση των σωματιδίων των υπολογισμένων διαφόρων τιμών του αερίου και των δεδομένων των επιφανειών του χώρου, ενώ παρουσιάζει περιγράμματα και διανυσματικές γραφικές παραστάσεις των στατικών δεδομένων οπουδήποτε μέσα σε ένα σκηνικό σε κάποιο συγκεκριμένο χρονικό σημείο.

Όλες οι παράμετροι εισαγωγής που απαιτούνται από το **FDS** για να περιγράψουν ένα συγκεκριμένο σενάριο εισάγονται στο πρόγραμμα μέσω ενός ή δύο αρχείων δεδομένων που δημιουργούνται από το χρήστη. Αυτά τα αρχεία περιέχουν πληροφορίες για το αριθμητικό πλέγμα, το περιβάλλον, τη γεωμετρία του κτιρίου, τις ιδιότητες των υλικών, την κινηματική της καύσης και τις επιθυμητές ποσότητες εξαγωγής. Το αριθμητικό πλέγμα είναι ένα ή περισσότερα ευθύγραμμα πλέγματα με (συνήθως) ομοιόμορφα κελιά. Όλα τα γεωμετρικά χαρακτηριστικά του σεναρίου πρέπει να προσαρμοστούν σε αυτό το αριθμητικό πλέγμα. Τα αντικείμενα που είναι μικρότερα από ένα κελί του πλέγματος είτε προσεγγίζονται ως ένα μονό κελί είτε απορρίπτονται. Η γεωμετρία των κτιρίων εισάγεται ως μία σειρά από ορθογώνια κουτιά και συνοριακές συνθήκες, που εφαρμόζονται στις ακραίες επιφάνειες. Τα υλικά καθορίζονται από τη θερμική αγωγιμότητά τους, ειδική θερμότητα, πυκνότητα, πάχος και συμπεριφορά καύσης. Υπάρχουν διάφοροι τρόποι εισαγωγής των πληροφοριών στο πρόγραμμα ανάλογα με

το επιθυμητό επίπεδο λεπτομέρειας. Ένα σημαντικό μέρος του αρχείου εισαγωγής του **FDS** κατευθύνει τον κώδικα να εξάγει διάφορες ποσότητες με διάφορους τρόπους. Όπως σε ένα πραγματικό πείραμα, ο χρήστης πρέπει να αποφασίσει προτού αρχίσουν οι υπολογισμοί, ποιες πληροφορίες επιθυμεί να αποθηκευτούν, καθώς δεν υπάρχει κανένας τρόπος να ανακτηθούν οι πληροφορίες αφότου τελειώσουν οι υπολογισμοί εάν αυτές δεν ζητήθηκαν στην αρχή.

Οποιαδήποτε προσομοίωση ενός σεναρίου πραγματικής πυρκαγιάς περιλαμβάνει τον ορισμό των ιδιοτήτων των υλικών για τους τοίχους, το πάτωμα, το ταβάνι και την επίπλωση. Το **FDS** αντιμετωπίζει όλα αυτά τα αντικείμενα ως ομοιογενή στερεά, κατά συνέπεια οι φυσικές παράμετροι για πολλά πραγματικά αντικείμενα μπορούν μόνο να αντιμετωπισθούν προσεγγιστικά βάσει των πραγματικών ιδιοτήτων τους. Η περιγραφή των υλικών στο αρχείο εισαγωγής είναι από μόνο του ένας προκλητικός στόχος για το χρήστη. Οι θερμικές ιδιότητες όπως η θερμική αγωγιμότητα, η ειδική θερμότητα, η πυκνότητα και το πάχος μπορούν να βρεθούν σε οδηγούς αναφοράς, σε κατασκευαστικά εγχειρίδια ή από πειραματικές μετρήσεις. Δυσκολότερη είναι η συμπεριφορά καύσης με διαφορετικές ροές θερμότητας, παρά το γεγονός ότι ολόκληρα βιβλία έχουν αφιερωθεί σε αυτό το θέμα, καθώς είναι δύσκολο να βρεθούν πληροφορίες για συγκεκριμένα στοιχεία.

1.1.3 Μοντέλα προσομοίωσης φωτιάς

Η ιδέα ότι η δυναμική μιας πυρκαγιάς μπορεί να μελετηθεί αριθμητικά έχει ερευνηθεί πολύ παλιά, από την εποχή των πρώτων υπολογισμών. Πράγματι, οι θεμελιώδεις εξισώσεις διατήρησης που περιγράφουν τη δυναμική ρευστών, η μεταφορά θερμότητας και η καύση γράφτηκαν αρχικά πριν από έναν αιώνα. Παρόλα αυτά, τα πρακτικά μαθηματικά μοντέλα της φωτιάς (σε διάκριση με την ελεγχόμενη καύση) είναι σχετικά πρόσφατα λόγω της έμφυτης πολυπλοκότητας του προβλήματος.

Οι δυσκολίες περιστρέφονται γύρω από κάποια ζητήματα: α) υπάρχει ένας τεράστιος αριθμός πιθανών σεναρίων πυρκαγιάς προς εξέταση, λόγω της τυχαίας φύσης τους, β) η επίγνωση της φυσικής του προβλήματος και η υπολογιστική δύναμη που απαιτούνται για την εκτέλεση όλων των απαραίτητων υπολογισμών για τα περισσότερα σενάρια πυρκαγιάς, είναι περιορισμένα. Μια ολοκληρωμένη μελέτη πυρκαγιάς πρέπει να εξετάσει τουλάχιστον μερικές πτυχές της αεροδυναμικής σωμάτων, της πολυφασικής ροής, της στροβιλώδους μίξης και καύσης, της ροής θερμότητας από ακτινοβολία και μεταγωγή, πεδία τα οποία είναι ενεργοί ερευνητικοί τομείς. Τέλος, γ) τα "καύσιμα" στις περισσότερες πυρκαγιές, δεν κατασκευάστηκαν για αυτό το σκοπό. Κατά συνέπεια, τα μαθηματικά μοντέλα και τα δεδομένα που χρειάζονται για την περιγραφή της φάσης της αποσύνθεσης των υλικών που παρέχουν τα καύσιμα, μπορεί

να μην είναι διαθέσιμα. Πράγματι, η μαθηματική μοντελοποίηση των φυσικών και χημικών μετασχηματισμών των πραγματικών υλικών καθώς καίγονται είναι ακόμα στα σπάργανα.

Προκειμένου να σημειωθεί πρόοδος, οι ερωτήσεις που υποβάλλονται πρέπει να απομακρυνθούν πολύ από την επιδίωξη μιας μεθοδολογίας που μπορεί να εφαρμοστεί σε όλα τα προβλήματα πυρκαγιάς. Ξεκινάμε με την εξέταση μερικών σεναρίων προς ανάλυση. Ενδεχομένως, οι μέθοδοι που αναπτύχθηκαν για να μελετήσουν αυτά τα "απλά" προβλήματα να μπορούν να γενικευτούν με την πάροδο του χρόνου έτσι ώστε τα πιο σύνθετα σενάρια να μπορούν να αναλυθούν. Έπειτα, πρέπει να μάθουμε να περιγράφουμε με ιδανικό τρόπο τις πυρκαγιές και τις κατά προσέγγιση λύσεις των εξιδανικευμένων εξισώσεων μας. Τέλος, οι μέθοδοι που δημιουργούμε πρέπει να επιδέχονται συστηματική βελτίωση. Καθώς η γνώση της φυσικής και η υπολογιστική μας δύναμη γίνεται ισχυρότερη, οι μέθοδοι ανάλυσης μπορούν να αναπτυχθούν μαζί τους.

Μέχρι σήμερα, τρεις διαφορετικές προσεγγίσεις έχουν προκύψει στην προσομοίωση των πυρκαγιών. Κάθε μια από αυτές τις αντιμετωπίζει τη πυρκαγιά ως μία εγγενώς τρισδιάστατη διαδικασία που εξελίσσεται σε συνάρτηση με το χρόνο. Η πρώτη που ωρίμασε, τα μοντέλα "ζώνης", περιγράφει τμήματα της φωτιάς. Κάθε τμήμα διαιρείται σε δύο χωρικά ομοιογενείς τομείς, ένα ζεστό ανώτερο στρώμα και ένα πιο δροσερό, χαμηλότερο στρώμα. Οι μαζικές και ενεργειακές ισορροπίες επιβάλλονται για κάθε στρώμα, με πρόσθετα πρότυπα, τα οποία περιγράφουν άλλες φυσικές διαδικασίες που έχουν προστεθεί σαν διαφορικές ή αλγεβρικές εξισώσεις. Παραδείγματα τέτοιων φαινομένων περιλαμβάνουν τις φλόγες της φωτιάς (fire plumes), τις ροές διαμέσου των πορτών, παραθύρων και άλλων διεξόδων, τη ροή της ακτινοβολούμενης και εκ μεταφοράς θερμότητας και την πυρόλυση στερεών καυσίμων. Οι περιγραφές των φυσικών και μαθηματικών υποθέσεων πίσω από την έννοια της μοντελοποίησης ζώνης δίνονται στις εργασίες των Jones [5] και Quintiere [6], τα οποία εξιστορούν τις εξελίξεις από το 1983. Η ανάπτυξη του μοντέλου από τότε έχει προχωρήσει στο σημείο όπου το τεκμηριωμένο και υποστηριζόμενο λογισμικό που υλοποιεί αυτό το μοντέλο, είναι ευρέως διαθέσιμο [7].

Η σχετικά φυσική και υπολογιστική απλότητα των μοντέλων ζώνης έχει οδηγήσει στη διαδεδομένη χρήση τους στην ανάλυση των σεναρίων πυρκαγιάς. Ως τώρα, εφόσον δεν απαιτείται λεπτομερής χωρική διανομή των σωματικών ιδιοτήτων και η περιγραφή δύο στρωμάτων προσεγγίζει λογικά την πραγματικότητα, αυτά τα μοντέλα είναι αρκετά αξιόπιστα. Εντούτοις, από την ίδια τη φύση τους, δεν υπάρχει κανένας τρόπος να βελτιωθούν συστηματικά. Η ταχεία ανάπτυξη της υπολογιστικής δύναμης και η αντίστοιχη ωρίμανση του τομέα της υπολογιστικής δυναμικής ρευστών (**CFD**), έχουν οδηγήσει στην ανάπτυξη των βασισμένων σε **CFD**, μοντέλα "τομέων" που χρησιμοποιήθηκαν σε ερευνητικά προβλήματα πυρκαγιάς. Ουσιαστικά όλη αυτά η εργασία είναι βασισμένη στον εννοιολογικό πλαίσιο που παρέχεται από τον Reynolds και τη μορφή των εξισώσεων των Navier-Stokes (**RANS**) και ιδιαίτερα το πρότυπο αναταραχής

k - ε από τους Patankar και Spalding [8]. Η χρήση των μοντέλων **CFD** έχει επιτρέψει την περιγραφή των πυρκαγιών σε σύνθετη γεωμετρία και την ενσωμάτωση μιας ευρείας ποικιλίας φυσικών φαινομένων. Εντούτοις, αυτά τα μοντέλα έχουν έναν θεμελιώδη περιορισμό για τις εφαρμογές πυρκαγιάς, τη μέση διαδικασία στη ρίζα των εξισώσεων του μοντέλου.

Τα μοντέλα **RANS** αναπτύχθηκαν ως ένας μέσου-χρόνου υπολογισμός στις εξισώσεις συντήρησης της δυναμικής ρευστών. Ενώ η ακριβής φύση του μέσου-χρόνου χρόνου δεν διευκρινίζεται, είναι σαφώς αρκετά μεγάλος για να γίνει η εισαγωγή των συντελεστών μεταφοράς μεγάλης δίνης/στροβίλου για την περιγραφή των ροών της μάζας, της ορμής και της ενέργειας. Αυτή είναι η πρωταρχική αιτία της εμφάνισης των αποτελεσμάτων ακόμη και των πιο ιδιαίτερα επιλυμένων προσομοιώσεων πυρκαγιάς. Οι μικρότερες επιλύσιμες κλίμακες μήκους καθορίζονται από το προϊόν της τοπικής ταχύτητας και το μέσο χρόνο παρά από τη χωρική διαμόρφωση του υπολογιστικού πλέγματος. Αυτή η ιδιότητα των μοντέλων **RANS** χρησιμοποιείται χαρακτηριστικά σε αριθμητικούς υπολογισμούς με τη χρήση αριθμητικών τεχνικών για να ληφθούν μεγάλα χρονικά βήματα.

Δυστυχώς, η εξέλιξη των μεγάλων δομών δίνης/στροβίλου, χαρακτηριστικών των περισσότερων φλογών της φωτιάς, χάνεται με μια τέτοια προσέγγιση, όπως είναι η πρόβλεψη των τοπικών παροδικών γεγονότων. Μερικές φορές υποστηρίζεται ότι η μέση διαδικασία που χρησιμοποιείται για να καθορίσει τις εξισώσεις είναι ένα ἐμέσο σύνολοη ανάμεσα σε πολλές εφαρμογές του ίδιου πειράματος ή ως αξίωμα. Εντούτοις, αυτό είναι ένα αμφισβητήσιμο σημείο στην έρευνα πυρκαγιάς, μιας και κανένα από τα πειράματα ή τα πραγματικά σενάρια δεν επαναλαμβάνονται υπό την έννοια που απαιτείται από εκείνη της ερμηνείας των εξισώσεων. Η εφαρμογή των τεχνικών "**Large Eddy Simulation**" (**LES**) στοχεύει στην εξαγωγή της μεγαλύτερης χρονικής και χωρικής πιστότητας από τις προσομοιώσεις της πυρκαγιάς που εκτελούνται στα πλέγματα, κάτι που καθίσταται δυνατό από τους ακόμα ταχύτερους υπολογιστές.

Η φράση **LES** αναφέρεται στην περιγραφή της ταραχώδους μίξης των αερίων και των προϊόντων καύσης με την τοπική ατμόσφαιρα που περιβάλλει την πυρκαγιά. Αυτή η διαδικασία, η οποία καθορίζει το ποσοστό καύσης στις περισσότερες πυρκαγιές και ελέγχει την διάδοση του καπνού και των καυτών αερίων, είναι εξαιρετικά δύσκολο να προβλεφθεί με ακρίβεια. Αυτό συμβαίνει όχι μόνο στην έρευνα μιας πυρκαγιάς αλλά σχεδόν σε όλα τα φαινόμενα που περιλαμβάνουν την ταραχώδη κίνηση ρευστών. Η βασική ιδέα πίσω από την τεχνική **LES** είναι ότι οι στρόβιλοι στους οποίους οφείλεται το μεγαλύτερο ποσοστό της μίξης είναι αρκετά μεγάλοι, ώστε να υπολογιστούν με λογική ακρίβεια από τις εξισώσεις της δυναμικής ρευστών. Η ελπίδα (που πρέπει τελικά να δικαιολογηθεί σε σύγκριση με τα πειράματα) είναι ότι η μικρής κλίμακας κίνηση στροβίλου μπορεί είτε να λογαριαστεί είτε να αγνοηθεί.

Οι εξισώσεις που περιγράφουν τη μεταφορά της μάζας, της ορμής και της ενέργειας από τις προκληθείσες από την πυρκαγιά, ροές πρέπει να απλοποιηθούν έτσι ώστε μπορούν να λυθούν αποτελεσματικά για τα σενάρια πυρκαγιάς που μας ενδιαφέρουν. Οι γενικές εξισώσεις της δυναμικής ρευστών περιγράφουν μια πλούσια ποικιλία φυσικών διαδικασιών, πολλές από τις οποίες δεν έχουν καμία σχέση με τις πυρκαγιές. Η διατήρηση αυτής της γενικότητας θα οδηγούσε σε μια πάρα πολύ σύνθετη υπολογιστική εργασία που θα έδινε πολύ λιγότερα στοιχεία της πυρκαγιάς. Οι απλουστευμένες εξισώσεις, που αναπτύχθηκαν από τους Rehm και Baum [9], έχουν ευρέως υιοθετηθεί από τη μεγαλύτερη ερευνητική κοινότητα μελέτης της καύσης, όπου αναφέρονται σε αυτές ως εξισώσεις καύσης "**low Mach number**". Αυτές περιγράφουν την αργή κίνηση ενός αερίου που οδηγείται από δυνάμεις χημικής απελευθέρωσης θερμότητας και πλευστότητας. Οι Oran και Boris παρέχουν μια χρήσιμη μελέτη της τεχνικής όπως εφαρμόζεται στα διάφορα αντιδραστικά καθεστώτα ροής στο κεφάλαιο με τίτλο "**Coupled Continuity Equations for Fast and Slow Flows**" [10]. Σχολιάζουν ότι "υπάρχει γενικά ένα μεγάλο κόστος για να είμαστε σε θέση να χρησιμοποιήσουμε έναν ενιαίο αλγόριθμο για τις γρήγορες αλλά και τις αργές ροές, ένα κόστος που μεταφράζεται σε πολλές υπολογιστικές πράξεις ανά χρονικό βήμα, το οποίο συχνά ξοδεύεται στην επίλυση των πολλαπλών και περίπλοκων πράξεων πινάκων".

Οι "**low Mach number**" εξισώσεις λύνονται αριθμητικά με τη διαίρεση του φυσικού χώρου όπου έχουμε την προσομοίωση της πυρκαγιάς με έναν μεγάλο αριθμό ορθογώνιων κελιών. Μέσα σε κάθε κελί η ταχύτητα του αερίου, η θερμοκρασία, κλπ., υποθέτουμε ότι είναι ομοιόμορφα κατανεμημένα και μεταβάλλονται μόνο με το χρόνο. Η ακρίβεια με την οποία η πυρκαγιά μπορεί να προσομοιωθεί εξαρτάται από τον αριθμό των κελιών που μπορούν να ενσωματωθούν στην προσομοίωση. Αυτός ο αριθμός είναι τελικά περιορισμένος από τη διαθέσιμη υπολογιστική δύναμη. Σήμερα, οι ηλεκτρονικοί υπολογιστές με έναν επεξεργαστή περιορίζουν τον αριθμό τέτοιου είδους κελιών το πολύ σε μερικά εκατομμύρια. Αυτό σημαίνει ότι οι κλίμακες μεγέθους της αναλογίας του μεγαλύτερου με τον μικρότερο στρόβιλο που μπορούν να επιλυθούν από τον υπολογισμό (το "δυναμικό πεδίο" της προσομοίωσης) είναι της τάξης του 100. Παράλληλη επεξεργασία μπορεί να χρησιμοποιηθεί για να επεκτείνει αυτό το πεδίο ως ένα ορισμένο βαθμό.

1.1.4 Χρήση του λογισμικού FDS

Στην περίπτωσή μας έχουμε υλοποιήσει τη συνολική δομή ενός δωματίου, μέσω της δημιουργίας ενός αρχείου εισόδου του **FDS**. Το αρχείο αυτό εμπεριέχει όλα τα δομικά στοιχεία του δωματίου, όπως έχουν αναφερθεί παραπάνω (οριοθέτηση δωματίου, ιδιότητες υλικών, επίπλωση, κλπ.) Με το δεδομένο αρχείο εισόδου και με τις κατάλληλες συνθήκες εκκίνησης της πυρκαγιάς,

το **FDS** παράγει κάποια αρχεία, τα οποία αποτελούν τα διάφορα σενάρια εξέλιξης της πυρκαγιάς σε συνάρτηση με το χρόνο.

Αξίζει εδώ να σημειωθεί ότι η παραγωγή των αρχείων εξόδου του **FDS** προϋποθέτει τη χρήση όχι μόνο ενός ηλεκτρονικού υπολογιστή, μιας και οι απαιτούμενοι υπολογισμοί είναι υπερβολικά πολλοί ακόμα και για ένα απλό σενάριο εξέλιξης μιας πυρκαγιάς και με τη χρήση μεγάλου χρονικού βήματος μεταξύ των υπολογισμών αυτών.

Για λόγους απλότητας θα χρησιμοποιήσουμε ένα από αυτά τα αρχεία ως βάση εξέλιξης της πυρκαγιάς. Σκοπός μας είναι η συλλογή των τρέχουσων θερμοκρασιών του χώρου του δωματίου (κάτι που επιτυγχάνεται από το δίκτυο αισθητήρων που είναι εγκατεστημένοι στο δωματιο) και η σύγκριση τους με τα στοιχεία του αρχείου εξόδου του **FDS**. Σε περίπτωση που έχουμε ταυτοποίηση των θερμοκρασιών αυτών σε κάποιο παράθυρο στο χρόνο, εμφανίζεται ένα πιθανό σενάριο. Κεντρική ιδέα αποτελεί το γεγονός ότι η εξέλιξη της πυρκαγιάς από το τρέχον χρονικό σημείο θα ακολουθήσει την ίδια πορεία με εκείνη του υπολογισμένου αρχείου. Με αυτό τον τρόπο γνωρίζουμε τι περίπου θα συμβεί ανάλογα με την επόμενη κίνηση που θα επιχειρήσουμε, όπως για παράδειγμα την εξέλιξη της πυρκαγιάς (τιμές θερμοκρασίας στα σημεία που υπάρχουν οι αισθητήρες) σε περίπτωση που ανοίξουμε μία πόρτα ή ένα παράθυρο του δωματίου. Έτσι μπορούμε να επιλέξουμε τις πιο ασφαλείς, για τα άτομα που επιχειρούν την κατάσβεση της φωτιάς, κινήσεις. Παραδείγμα ερώτησης μπορεί να είναι το παρακάτω:

-Είναι ασφαλές να ανοιχτεί η πόρτα του δωματίου στα επόμενα 5 λεπτά;

Εφόσον έχουμε ταυτοποιήσει την τρέχουσα κατάσταση με κάποιο σενάριο, τότε μπορούμε να δούμε την εξέλιξη των κατά τόπους θερμοκρασιών, στην περίπτωση που ανοιχτεί η πόρτα. Μελετούμε το ζητούμενο χρονικό διάστημα και δεδομένου των προκαθορισμένων χαρακτηριστικών ασφαλείας, λαμβάνουμε την κατάλληλη απόφαση. Αν για παράδειγμα, δούμε ότι η εξέλιξη των θερμοκρασιών στο δωμάτιο αυξάνεται δραματικά στο άμεσο χρονικό μέλλον, τότε η απόφαση να ανοίξει η πόρτα πρέπει να αποτραπεί, ώστε να μην κινδυνεύσουν οι πυροσβέστες.

1.2 Jade (Java Agent DEvelopment Framework)

1.2.1 Εισαγωγή

Το **JADE** (**Java Agent DEvelopment Framework**) είναι λογισμικό, υλοποιημένο πλήρως σε **Java**. Απλοποιεί την υλοποίηση συστημάτων πολυ-πρακτόρων μέσω ενός **middle-ware**[1] που συμμορφώνεται με τις προδιαγραφές **FIPA** και μέσω ενός συνόλου εργαλείων που υποστηρίζει τις φάσεις διόρθωσης (**debugging**) και επέκτασης. Η πλατφόρμα πρακτόρων μπορεί να κατανεμηθεί στα διάφορα μηχανήματα (τα οποία δεν απαιτείται να μοιράζονται το ίδιο λειτουργικό σύστημα) και η διαμόρφωση μπορεί να ελεγχθεί μέσω ενός απομακρυσμένου γραφικού περιβάλλοντος χρήστη (**remote GUI**). Η διαμόρφωση μπορεί να αλλάξει ακόμη και κατά το χρόνο εκτέλεσης με την μετακίνηση των πρακτόρων από το ένα μηχάνημα σε κάποιο άλλο, όταν και όπως ζητηθεί. Το **JADE** είναι υλοποιημένο σε **Java** και οι ελάχιστες απαιτήσεις του συστήματος είναι η έκδοση 1.4 της **Java** (**run time environment** ή **JDK**, που τώρα καλείται **SDK**).

Η σύμπραξη μεταξύ της πλατφόρμας του **JADE** και των βιβλιοθηκών **LEAP** επιτρέπει τη δημιουργία μιας πλατφόρμας πρακτόρων συμμορφωμένη με την **FIPA** με μειωμένες απαιτήσεις και συμβατής με τα περιβάλλοντα **Java** των κινητών τηλεφώνων έως το **J2ME-CLDC MIDP 1.0**. Οι βιβλιοθήκες **LEAP** έχουν αναπτυχθεί με τη συνεργασία του προγράμματος **LEAP** και μπορούν να μεταφορτωθούν ως πρόσθετο στοιχείο του **JADE** από τον ίδιο ιστοχώρο.

Το **JADE** είναι λογισμικό ελεύθερης χρήσης και διανέμεται από το **TILAB**, τον κάτοχο των πνευματικών δικαιωμάτων, σαν **open source** λογισμικό υπό τους όρους του **LGPL** (**Lesser General Public License Version 2**). Από τον Μάιο του2003, έχει δημιουργηθεί ένα σύνολο εταιριών (**JADE Board**) που επιβλέπει τη διαχείριση του προγράμματος **JADE**. Αυτήν την περίοδο, το σύνολο αυτό απαριθμεί 5 μέλη: **Telecom Italia**, **Motorola**, **Whitestein Technologies AG**, **Profactor GmbH** και **France Telecom R&D**.

Η πιο πρόσφατη έκδοση του **JADE** είναι η έκδοση **JADE 3.5** που δόθηκε προς χρήση στις 25 Ιουνίου 2007.

Το **JADE** είναι ένα **middle-ware** που στοχεύει στην υποστήριξη της ανάπτυξης εφαρμογών που εξετάζουν αυτή την εξέλιξη, καθώς είναι βασισμένο στην ευφυή, **peer-to-peer** προσέγγιση πρακτόρων.

[1] Λογισμικό που συνδέει διαφορετικές συνιστώσες από ανομοιογενή πολύπλοκα λογισμικά πακέτα για σωστή μεταφορά των δεδομένων σε κατανεμημένες εφαρμογές

Το **JADE** επιτρέπει την ανάπτυξη συστημάτων με **peers** ικανούς για να:

- εργάζονται με δυναμικό τρόπο, σύμφωνα με τους κανόνες χρήσης που δίνονται από τους ιδιοκτήτες

- επικοινωνούν και να διαπραγματεύονται με άλλους **peers**, άμεσα και ανεξάρτητα από το ρόλο και τη θέση τους

- συντονιστούν προκειμένου να λύσουν κάποια σύνθετα προβλήματα με έναν κατανεμημένο τρόπο

Οι τομείς εφαρμογής όπου τα συστήματα του **JADE** μπορούν να σχεδιαστούν αποτελεσματικά και να επεκταθούν είναι σχεδόν όλοι, από τις υπηρεσίες διαδικτύου που απευθύνονται στον καταναλωτή ή σε εταιρίες μέχρι το περιβάλλον του κινητού και την κάλυψη του, με τα βέλτιστα αποτελέσματα, καθώς επίσης και τον τομέα των εφαρμογών μηχανής-προς-μηχανή (**machine-to-machine**), όπου η δυναμικότητα των **peers** και ο συντονισμός της κατανεμημένης εργασίας είναι λειτουργίες που ταιριάζουν τέλεια στις απαιτήσεις τυπικών αυτοματοποιημένων δικτύων.

Επίσης, όλα αυτά τα περιβάλλοντα μπορούν να μοιραστούν το ίδιο βασικό σύνολο λειτουργιών που απαιτούνται για την υποστήριξη των πρωτοκόλλων επικοινωνίας, των αυτόνομων συμπεριφορών και της διαχείρισης των **peers**.

Το **JADE** υλοποιεί ένα βασισμένο σε **Java middleware** που συλλέγει όλες αυτές τις δομικές μονάδες και προσφέρει ένα εκτενές σύνολο από **API's** (**Application Programming Interface**), παράλληλα με μια άριστη σειρά εργαλείων προγραμματισμού και δοκιμής για εύκολη και διαισθητική ανάπτυξη των εφαρμογών που βασίζονται σε **p2p** πράκτορες.

Επιπλέον, η μεγάλη δύναμη του **JADE** είναι η φορητότητά του σε διαφορετικά περιβάλλοντα κατά τη διάρκεια της εκτέλεσης: χάρη στη συμβολή του προγράμματος με όνομα **LEAP**, του ευρωπαϊκού **IST**, το **JADE** έχει εκδόσεις για κάθε προφίλ των μηχανών της **Java**, από **J2EE** για τους κεντρικούς υπολογιστές διαδικτύου (**Internet servers**) έως το **J2ME MIDP** για τα τερματικά κινητών.

Το **JADE** έχει συλληφθεί σαν ιδέα και έχει αναπτυχθεί από το **Tilab**, το **R&D** τμήμα της **Telecom Italia**, αλλά από την αρχή έχει επιλεχθεί ένας ανοικτός τρόπος για την πρόταση και την επέκταση του. Το **JADE** τίθεται στην διάθεση του καθενός με άδεια του **LGPL** και έχει δημιουργηθεί μια κοινότητα ανοικτού κώδικα γύρω από αυτό (**Open Source Community**).

1.2.2 Τεχνική περιγραφή

Ο στόχος του **JADE** είναι να απλοποιηθεί η ανάπτυξη των συστημάτων πολυ-πρακτόρων εξασφαλίζοντας τυποποιημένη συμμόρφωση μέσω ενός περιεκτικού συνόλου υπηρεσιών συστήματος και πρακτόρων, σύμφωνα με τις προδιαγραφές **FIPA**. Η πλατφόρμα πρακτόρων του **JADE** συμμορφώνεται με τις προδιαγραφές **FIPA** και περιλαμβάνει όλα εκείνα τα υποχρεωτικά συστατικά που διαχειρίζονται την πλατφόρμα, τα οποία είναι τα: **ACC**, **AMS** και **DF**. Όλες οι επικοινωνίες μεταξύ πρακτόρων εκτελούνται μέσω της διαβίβασης μηνυμάτων, όπου η **FIPA ACL** είναι η γλώσσα που αντιπροσωπεύει τα μηνύματα. Η πλατφόρμα των πρακτόρων μπορεί να κατανεμηθεί σε διάφορους οικοδεσπότες (**hosts**). Μόνο μια εφαρμογή της **Java** και επομένως μόνο μια εικονική μηχανή της **Java** (**JVM**) εκτελείται σε κάθε **host**. Κάθε **JVM** βασικά περιλαμβάνει τους πράκτορες και παρέχει ένα πλήρες περιβάλλον χρόνου εκτέλεσης για την εκτέλεση των πρακτόρων και επιτρέπει σε διάφορους πράκτορες να εκτελούνται ταυτόχρονα στον ίδιο **host**.

Η αρχιτεκτονική επικοινωνίας προσφέρει ευέλικτα και αποδοτικά μηνύματα, όπου το **JADE** δημιουργεί και διαχειρίζεται μια ουρά από εισερχόμενα μηνύματα ACL, ιδιωτικά για κάθε πράκτορα. Οι πράκτορες μπορούν να έχουν πρόσβαση στη ουρά αναμονής τους μέσω ενός συνδυασμού διάφορων τρόπων, βασισμένων σε: **blocking**, **polling**, **timeout** και **pattern matching**. Το πλήρες πρότυπο επικοινωνίας **FIPA** έχει υλοποιηθεί και τα συστατικά του έχουν σαφώς διακριθεί και ενσωματωθεί πλήρως: πρωτόκολλα αλληλεπίδρασης, **envelope**, **ACL**, γλώσσες περιεχομένου, σχήματα κωδικοποίησης, οντολογίες και τελικά πρωτόκολλα μεταφοράς. Ο μηχανισμός μεταφοράς ειδικότερα, έχει χαρακτηριστικά χαμαιλέοντα, μιας και προσαρμόζεται σε κάθε κατάσταση, επιλέγοντας με διαφάνεια το καλύτερο διαθέσιμο πρωτόκολλο. Προς το παρόν χρησιμοποιούνται τα πρωτόκολλα **Java RMI**, **event-notification**, **HTTP** και **IIOP**, αλλά μπορούν να προστεθούν εύκολα περισσότερα πρωτόκολλα μέσω των διεπαφών **MTP** και **IMTP** του **JADE**. Τα περισσότερα από τα πρωτόκολλα αλληλεπίδρασης που καθορίζονται από την **FIPA** είναι ήδη διαθέσιμα και μπορούν να χρησιμοποιηθούν μετά από τον καθορισμό της, εξαρτημένης από την εφαρμογή, συμπεριφοράς για κάθε κατάσταση του πρωτοκόλλου. Έχουν υλοποιηθεί ήδη τα πρωτόκολλα **SL** και **agent management ontology**, καθώς επίσης και η υποστήριξη για τις, καθορισμένες από το χρήστη, γλώσσες περιεχομένου και τις οντολογίες που μπορούν να υλοποιηθούν, να καταχωρηθούν με τους πράκτορες και να χρησιμοποιηθούν αυτόματα από το πλαίσιο.

Βασικά, οι πράκτορες υλοποιούνται με τον περιορισμό ενός νήματος (**thread**) ανά πράκτορα, αλλά οι πράκτορες πρέπει συχνά να εκτελέσουν παράλληλες εργασίες. Σε συνέχεια της λύσης

μέσω χρήσης πολλών νημάτων, που προσφέρεται άμεσα από τη **Java**, το **JADE** υποστηρίζει επίσης τον σχεδιασμό συμπεριφορών που συνεργάζονται, όπου το **JADE** προγραμματίζει αυτές τις εργασίες με έναν ελαφρύ και αποδοτικό τρόπο. Ο χρόνος εκτέλεσης περιλαμβάνει επίσης μερικές έτοιμες προς χρήση συμπεριφορές για τις πιο κοινές εργασίες στον προγραμματισμό πρακτόρων, όπως τα πρωτόκολλα αλληλεπίδρασης **FIPA**, η ενεργοποίηση υπό μία ορισμένη συνθήκη και η δόμηση σύνθετων εργασιών ως άθροισμα απλούστερων. Μεταξύ άλλων, το **JADE** προσφέρει επίσης την αποκαλούμενη **JessBehaviour** που επιτρέπει την πλήρη ολοκλήρωση με το **JESS**, όπου το **JADE** παρέχει το **shell** του πράκτορα και εγγυάται (όπου είναι δυνατόν) τη συμμόρφωση κατά **FIPA**, ενώ το **JESS** είναι η μηχανή του πράκτορα που εκτελεί όλο τον απαραίτητο συλλογισμό.

Η πλατφόρμα των πρακτόρων παρέχει ένα γραφικό περιβάλλον χρήστη (**GUI**) για απομακρυσμένη διαχείριση (**remote management**), παρακολούθηση και έλεγχο της κατάστασης των πρακτόρων, επιτρέποντας παραδείγματος χάριν, να σταματήσει και να επανεκκινήσει τους πράκτορες. Το **GUI** επιτρέπει επίσης τη δημιουργία και την εκκίνηση της εκτέλεσης ενός πράκτορα σε έναν απομακρυσμένο **host**, υπό τον όρο ότι ένα **agent container** τρέχει ήδη. Το **GUI** επιτρέπει επίσης τον έλεγχο άλλων απομακρυσμένων, συμμορφωμένων κατά **FIPA**, πλατφορμών πρακτόρων.

Μέσα στον πυρήνα του **JADE** έχουν υλοποιηθεί διάφορα γραφικά εργαλεία που υποστηρίζουν τη φάση διόρθωσης, κάτι συνήθως αρκετά σύνθετο σε κατανεμημένα συστήματα. Το **Dummy Agent** είναι ένα απλό, όμως πολύ χρήσιμο, εργαλείο για την επίβλεψη των μηνυμάτων που ανταλλάσσονται μεταξύ των πρακτόρων. Το γραφικό περιβάλλον παρέχει την υποστήριξη για την διόρθωση, τη σύνθεση και αποστολή των **ACL** μηνυμάτων προς τους πράκτορες,τη λήψη και εμφάνιση των μηνυμάτων από τους πράκτορες και τελικά την αποθήκευση (**save**) και την φόρτωση (**load**) των μηνυμάτων από και προς το δίσκο.

Το **Sniffer Agent**, επιτρέπει τον εντοπισμό των μηνυμάτων που ανταλλάσσονται σε μια πλατφόρμα πρακτόρων του **JADE**. Όταν ο χρήστης αποφασίζει να παρακολουθήσει (**sniff**) έναν πράκτορα ή μια ομάδα πρακτόρων, κάθε μήνυμα που κατευθύνεται ή προέρχεται από εκείνο τον πράκτορα ή την ομάδα, παρακολουθείται και επιδεικνύεται στο ενδεικτικό παράθυρο του εργαλείου. Ο χρήστης μπορεί να δει, να αποθηκεύσει και να φορτώσει κάθε διαδρομή μηνυμάτων για την πιο πρόσφατη ανάλυση.

Το **Introspector Agent**, επιτρέπει την επιτήρηση και τον έλεγχο του κύκλου ζωής ενός ενεργού πράκτορα και των ανταλλαγμένων μηνυμάτων του στη ουρά αναμονής των απεσταλμένων και λαμβανόμενων μηνυμάτων του.

1.2.3 Χρήση του λογισμικού JADE

Το σύνολο της υλοποίησης βασίζεται στις λειτουργίες τεσσάρων πρακτόρων του **Jade**. Οι τέσσερεις αυτοί πράκτορες είναι οι: **ServerAgent**, **GatewayAgent**, **CompareAgent**, και **pdaAgent** και κάθε ένας από αυτούς συμβάλλει στην ορθή λειτουργία του προγράμματος μας.

Ειδικότερα, ο πράκτορας **ServerAgent**, όπως λέει και το όνομά του, αναλαμβάνει το ρόλο του **Server**. Δηλαδή, όταν δεχτεί ένα κατάλληλο μήνυμα (**ACLMessage** με τιμή του **String query = "StartServer"**), ανοίγει ένα **Socket** στο **port** 4444 και αναμένει κάποιον **Client** να συνδεθεί, για να του μεταδώσει τις τιμές θερμοκρασίας του δικτύου αισθητήρων. Η σύνδεση των **server** και **client** πραγματοποιείται μέσω μιας **TCP** σύνδεσης. Τέλος, όταν δεχτεί ένα κατάλληλο μήνυμα (**ACLMessage** με τιμή του **String query = "StopServer"**), ο πράκτορας τερματίζει την λειτουργία του ως **server**, κλείνοντας το **Socket**.

Ο δεύτερος πράκτορας είναι ο **GatewayAgent**, ο οποίος εκτελεί τρεις λειτουργίες:

1. Αρχικά, εκτελεί χρέη **Client**, δηλαδή, αναμένει ένα κατάλληλο μήνυμα (**ACLMessage** με τιμή του **String query = "StartClient"**) και όταν το λάβει, συνδέεται με τον **Server** στο **port** 4444 και ξεκινά να λαμβάνει και να αποθηκεύει σε κατάλληλη δομή, τις τιμές θερμοκρασίας των αισθητήρων. Ακόμα, όταν δεχτεί κατάλληλο μήνυμα (**ACLMessage** με τιμή του **String query = "StopClient"**), ο πράκτορας τερματίζει την λειτουργία του ως **client**, δηλαδή αποσυνδέεται από τον **server**, σταματά να αποθηκεύει τιμές θερμοκρασίας και μέσω της συνάρτησης **save_current_data** αποθηκεύει το ιστορικό των μετρήσεων σε ένα αρχείο με όνομα **sensor_data.txt** μαζί με την τρέχουσα ημερομηνία/ώρα.

2. Για την περίπτωση που κάποιος πράκτορας θέλει τις τρέχουσες τιμές θερμοκρασίας από το δίκτυο των αισθητήρων, αποστέλλει στον πράκτορα **GatewayAgent** ένα **ACLMessage** με τιμή του **String query = "GiveNowValues"**. Τότε, αφού συλλεχθούν οι απαραίτητες πληροφορίες, θα ταξινομηθούν κατά **sensor_id** και θα σταλούν στον πράκτορα που τις ζήτησε.

3. Τέλος, όμοια με παραπάνω, στην περίπτωση που κάποιος πράκτορας θέλει τις τιμές θερμοκρασίας από το δίκτυο των αισθητήρων για κάποιο συγκεκριμένο χρονικό διάστημα (σε δευτερόλεπτα), αποστέλλει στον πράκτορα **GatewayAgent** ένα **ACLMessage** με τιμή του **String query = "GiveContValues"**. Με όμοιο τρόπο, ο πράκτορας θα συλλέξει τις απαραίτητες πληροφορίες, θα τις ταξινομήσει κατά **sensor_id** και θα τις στείλει στον πράκτορα που τις ζήτησε.

Ο τρίτος πράκτορας είναι ο **CompareAgent**, ο οποίος κατά κάποιο τρόπο οργανώνει και διαχειρίζεται τη λειτουργία των προηγούμενων δύο. Ουσιαστικά ο πράκτορας αυτός μπορεί να διαμορφωθεί ανάλογα με τις απαιτούμενες συνθήκες χρήσης του. Στην περίπτωση μας, όταν αρχικοποιείται "φορτώνει" τα απαιτούμενα δεδομένα και κατόπιν στέλνει ένα μήνυμα προς τον πράκτορα **ServerAgent**, ώστε να ξεκινήσει τη λειτουργία του ως **server** και ένα μήνυμα προς τον πράκτορα **GatewayAgent**, ώστε να ξεκινήσει τη λειτουργία του ως **client**. Στη συνέχεια ο πράκτορας αναμένει μηνύματα εξυπηρέτησης τύπου **ACLMessage** από πράκτορες που εκτελούνται σε περιβάλλον **pda** και στην δική μας περίπτωση πιο συγκεκριμένα από ένα "αντικείμενο" του πράκτορα **pdaAgent**. Όταν κάποιο μήνυμα καταφτάσει, διαβάζονται τα στοιχεία του και ανάλογα με το περιεχόμενό του εκτελεί τις εξής ενέργειες:

- τιμή **String = "getScenario"**: αρχικά ζητά από τον πράκτορα **GatewayAgent** τις τρέχουσες θερμοκρασίας των αισθητήρων. Στη συνέχεια καλεί τη συνάρτηση **search_patt**, η οποία υπολογίζει μέσω των διαθέσιμων συναρτήσεων της ένα πιθανό σενάριο εξέλιξης της πυρκαγιάς και στέλνει το αποτέλεσμα στον πράκτορα που απέστειλε το ερώτημα.

- τιμή **String = "getExtendedScenario"**: αρχικά ζητά από τον πράκτορα **GatewayAgent** τις τιμές θερμοκρασίας των αισθητήρων για χρονικό διάστημα **dt**. Στη συνέχεια καλεί τη συνάρτηση **search_patt**, η οποία υπολογίζει μέσω των διαθέσιμων συναρτήσεων της ένα πιθανό σενάριο εξέλιξης της πυρκαγιάς και στέλνει το αποτέλεσμα στον πράκτορα που απέστειλε το ερώτημα.

- τιμή **String = "exit"**: σταματά τις λειτουργίες **client/server** (αποστέλλοντας κατάλληλα μηνύματα) και στη συνέχεια τερματίζει τη λειτουργία του.

Ο τελευταίος πράκτορας που εμπλέκεται είναι ο **pdaAgent**, ο οποίος είναι ο πράκτορας που θα "τρέχει" στην φορητή συσκευή. Για να είναι αυτό δυνατό, στην φορητή συσκευή θα τρέχει η έκδοση **Jade Leap** του **Jade**, η οποία είναι αρκετά ελαφριά (μικρή κατανάλωση διαθέσιμων πόρων).

Αφού εμφανιστεί το μήνυμα ετοιμότητας του πράκτορα, ακολουθεί μια μικρή παύση, για να δώσουμε λίγο χρόνο στον **client**, ώστε να συλλέξει κάποιες τιμές από το δίκτυο των αισθητήρων. Στη συνέχεια εμφανίζεται στο χρήστη το κεντρικό μενού (**main menu**) με τις επιλογές:

1. **Get possible scenario**
2. **Exit**

Οι επιλογές στο κεντρικό μενού και στα διάφορα υπομενού πραγματοποιούνται μέσω πληκτρολογίου, σύμφωνα με τις διαθεσιμες επιλογές που εμφανίζονται.

Έτσι, ανάλογα με την επιλογή μας, θα έχουμε:

1. Με αυτή την επιλογή, ο **pdaAgent** θα ζητήσει από τον **CompareAgent** ένα πιθανό σενάριο εξέλιξης της πυρκαγιάς. Ανάλογα με το αποτέλεσμα που θα επιστραφεί εμφανίζεται κάποιο σχετικό υπομενού με τις κατάλληλες επιλογές.

   - **Δεν υπάρχει κάποιο αποτέλεσμα**: τότε εμφανίζεται ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to main menu**, όπου μας επιστρέφει στο κεντρικό μενού.

   - **Υπάρχει ένα αποτέλεσμα**: τότε εμφανίζεται το αποτέλεσμα και επιπλέον μας δίνεται η επιλογή σχετικά με το αν επιθυμούμε να παρακολουθήσουμε το σχετικό βίντεο μέσω του **Smokeview** ή όχι. Στη συνέχεια εμφανίζεται ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to main menu**, όπου μας επιστρέφει στο κεντρικό μενού.

   - **Υπάρχουν περισσότερα από ένα αποτελέσματα**: τότε εμφανίζεται το μήνυμα **"More than one, possible scenarios found (x)! What do you want to do?"** (όπου **x** το πλήθος των σεναρίων που βρέθηκαν) και ένα υπομενού με τις επιλογές **Show all possible scenarios**, όπου τυπώνει όλα τα πιθανά σενάρια και επιστρέφει στο κεντρικό μενού, **Exit to main menu**, όπου μας επιστρέφει στο κεντρικό μενού και **Request more values from network**, όπου ζητά από τον πράκτορα **compareAgent** ένα πιθανό σενάριο με βάση τις τιμές θερμοκρασίας για χρονικό διάστημα **dt**. Αν επιλέξουμε **Request more values from network**, ανάλογα με την απάντησή του **CompareAgent**, έχουμε:

     - **δεν υπάρχει κάποιο αποτέλεσμα**: τότε εμφανίζεται το σχετικό μήνυμα και ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to previous menu**, όπου μας επιστρέφει στο προηγούμενο υπομενού.

     - **υπάρχει ένα αποτέλεσμα**: τότε εμφανίζεται το αποτέλεσμα και επιπλέον μας δίνεται η επιλογή σχετικά με το αν επιθυμούμε να παρακολουθήσουμε

το σχετικό βίντεο μέσω του **Smokeview** ή όχι. Στη συνέχεια εμφανίζεται ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to previous menu**, όπου μας επιστρέφει στο κεντρικό μενού.

- **υπάρχουν περισσότερα από ένα αποτελέσματα**: τότε εμφανίζεται το μήνυμα **"More than one, possible scenarios found (x)! What do you want to do?"** (όπου **x** το πλήθος των σεναρίων που βρέθηκαν) και το σχετικό υπομενού όπως αυτό έχει περιγραφεί παραπάνω.

2. Με αυτή την επιλογή, ο **pdaAgent** θα τερματίσει την λειτουργία του και θα ειδοποιήσει τον **CompareAgent**, ώστε να σταματήσει και αυτός με τη σειρά του, τους **client/server**.

Κεφάλαιο 2

Ένα απλό πλαίσιο πύλης (gateway) για δίκτυα αισθητήρων

2.1 Εισαγωγή

Η πρόσφατη πρόοδος στις τηλεπικοινωνίες, τα ενσωματωμένα συστήματα (**embedded systems**) και την τεχνολογία αισθητήρων καθιστούν τα ασύρματα δίκτυα αισθητήρων μια ελκυστική εναλλακτική λύση στις ενσύρματες δομές για την υποστήριξη ποικίλων εφαρμογών παρακολούθησης και ελέγχου δραστηριότητας. Το κύριο πλεονέκτημά τους είναι η ευέλικτη εγκατάσταση χωρίς την ανάγκη να χρησιμοποιηθούν καλώδια, κάτι το οποίο μπορεί στη συνέχεια να απλοποιήσει την επέκταση και να μειώσει το κόστος. Έχει πραγματοποιηθεί εργασία στα συστήματα χρόνου εκτέλεσης και **middleware** σε μία προσπάθεια να απλοποιηθεί η ανάπτυξη, η εγκατάσταση και η εκτέλεση του εφαρμοζόμενου κώδικα στους κόμβους. Οι εφαρμογές πελατών (**Client**) βλέπουν ένα δίκτυο αισθητήρων ως πηγή πληροφοριών. Είναι λοιπόν σημαντικό να παρασχεθούν μηχανισμοί που επιτρέπουν και διευκολύνουν την πρόσβαση στις πληροφορίες αυτές με έναν ευέλικτο και αποδοτικό τρόπο. Μεταξύ των θεμάτων που προκύπτουν, τα ακόλουθα δύο είναι ιδιαίτερα σημαντικά:

1. Οι **clients** που επιθυμούν να έχουν πρόσβαση στο δίκτυο αισθητήρων θα κατοικήσουν πιθανότατα σε υπολογιστές που δεν έχουν υποστήριξη για το απαραίτητο υλικό και πρωτόκολλο επικοινωνίας του δικτύου αισθητήρων. Ως εκ τούτου, πρέπει να παρασχεθεί λειτουργικότητα της πύλης, και στο φυσικό επίπεδο δικτύου και στο επίπεδο πρωτοκόλλου, καθιστώντας την πρόσβαση στις πληροφορίες όσο το δυνατόν διαφανέστερη για τον πελάτη.

2. Οι διάφοροι πελάτες μπορεί να θέλουν να έχουν πρόσβαση σε ένα δίκτυο αισθητήρων ταυτόχρονα. Λαμβάνοντας υπόψη τους χαρακτηριστικούς περιορισμούς των πόρων των

δικτύων αισθητήρων, κάποια αιτήματα πρέπει να παρεμποδιστούν και να συνδυαστούν μαζί στην άκρη του δικτύου αντί να ωθηθούν αφελώς στο δίκτυο.

Θα παρουσιαστεί ένα πλαίσιο για την υλοποίηση ενός ασύρματου δικτύου αισθητήρων σε ένα κατανεμημένο υπολογιστικό περιβάλλον, το οποίο προσπαθεί να εξετάσει αυτές τις απαιτήσεις.

2.2 Το πλαίσιο SensorNetGateway

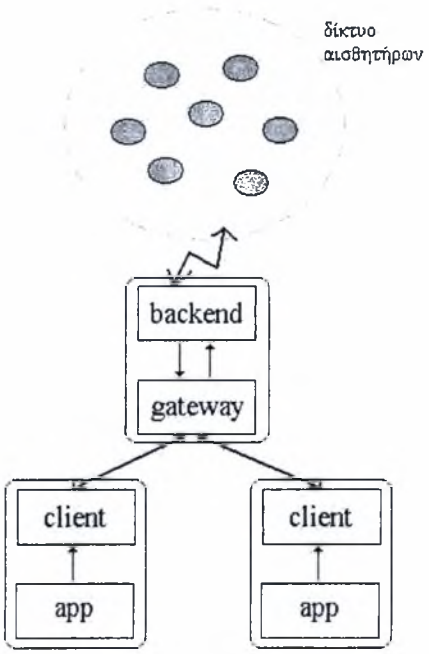
Το **SensorNetGatewayFramework** (**SNG**) είναι ένα σύστημα λογισμικού που έχει αναπτυχθεί για να επιτρέψει την ενσωμάτωση ενός δικτύου αισθητήρων σε ένα κατανεμημένο υπολογιστικό περιβάλλον. Τα προγράμματα πελατών μπορούν να εκμεταλλευτούν το **SNG** με στόχο την πρόσβαση και τον έλεγχο ενός δικτύου αισθητήρων ανεξάρτητα από την εσωτερική του υλοποίηση, όπως π.χ. το υλικό (**hardware**), την πλατφόρμα που "τρέχει" και το πρωτόκολλο επικοινωνίας. Ένα δίκτυο αισθητήρων μπορεί να επεκταθεί μέσω του **SNG** χωρίς να πρέπει να εξεταστεί η ταυτόχρονη πρόσβαση των πελατών και η συγκέντρωση στην περιφέρεια του δικτύου. Το **SNG** είναι υλοποιημένο σε **Java**, κυρίως για να απλοποιηθεί η ανάπτυξη αλλά και λόγω της φορητότητας για τις διαφορετικές πλατφόρμες, συμπεριλαμβανομένων των κινητών συσκευών. Το **SNG** περιγράφεται λεπτομερέστερα παρακάτω.

2.3 Αρχιτεκτονική

Η αρχιτεκτονική του **SNG**, η οποία φαίνεται στο **Σχήμα** 2.1, περιλαμβάνει τα ακόλουθα τμήματα λογισμικού:

- Το τμήμα των κόμβων των αισθητήρων αντιπροσωπεύει το πρόγραμμα που κατοικεί σε κάθε κόμβο του δικτύου αισθητήρων. Εκτελεί τις μετρήσεις των αισθητήρων μεταδίδει τις πληροφορίες στο οπίσθιο μέρος (**backend**) του δικτύου των αισθητήρων. Προφανώς, είναι επίσης υπεύθυνο για την υλοποίηση του εσωτερικού πρωτοκόλλου του δικτύου αισθητήρων, σύμφωνα με την επιθυμητή λειτουργικότητα.

- Το τμήμα οπίσθιου μέρους (**backend**) του δικτύου αισθητήρων είναι υπεύθυνο για την πραγματική επικοινωνία με το δίκτυο των αισθητήρων, για λογαριασμό της πύλης. Διαδίδει τα αιτήματα που διανέμονται από το τμήμα της πύλης στους κόμβους αισθητήρων



Σχήμα 2.1: Αρχιτεκτονική SNG

και λαμβάνει τα πακέτα που στέλνονται από τους κόμβους αισθητήρων και τα παραδίδει στο τμήμα της πύλης.

- Το τμήμα της πύλης του δικτύου αισθητήρων εφαρμόζει τη γενική λειτουργία πυλών, η οποία είναι ανεξάρτητη από την τεχνολογία που χρησιμοποιείται για την υλοποίηση του δικτύου αισθητήρων. Παρεμποδίζει και αθροίζει τα αιτήματα πελατών προκειμένου αυτά να διαδοθούν στο δίκτυο αισθητήρων μέσω του οπίσθιου μέρους του δικτύου αισθητήρων. Αντιθέτως, λαμβάνει τα δεδομένα των κόμβων από το οπίσθιο μέρος του δικτύου αισθητήρων και τα προωθεί στους αντίστοιχους πελάτες.

- Τέλος, το τμήμα πελατών του δικτύου αισθητήρων είναι υπεύθυνο για την αλληλεπίδραση με το τμήμα πύλης μέσω του δικτύου υπό μορφή ενός κατάλληλου **API**. Οι προγραμματιστές εφαρμογών έτσι δεν πρέπει να ανησυχήσουν για το γεγονός ότι το δίκτυο αισθητήρων μπορεί να επεκταθεί σε μια απομακρυσμένη τοποθεσία ή ότι χρησιμοποιεί ιδιόκτητο πρωτόκολλο και υλικό (**hardware**).

Η πύλη και ο πελάτης του δικτύου αισθητήρων είναι υλοποιημένα μέρη του πλαισίου, ενώ ο κόμβος του αισθητήρα και το οπίσθιο μέρος του δικτύου αισθητήρων πρέπει να υλοποιηθούν σύμφωνα με τις προδιαγραφές του δικτύου αισθητήρων που θα αναπτυχθεί. Το οπίσθιο μέρος και η πύλη πρέπει να κατοικήσουν σε ένα μηχάνημα που είναι σε θέση να επικοινωνήσει άμεσα με (τουλάχιστον έναν από) τους κόμβους των αισθητήρων. Από άποψη η δομή τεχνολογίας λογισμικού, η αρχιτεκτονική του SNG είναι ένας συνδυασμός των μοντέλων **"model-view-control"** και **"bridge"/"delegator"**.

2.4 Διεπαφές (Interfaces) προγραμματισμού

Οι διεπαφές **SNG** παρέχουν απλές, καθαρές και αφαιρετικές έννοιες επικοινωνίας στον υπεύθυνο για την ανάπτυξη, επιτρέποντας ταυτόχρονα μια αποδοτική υλοποίηση του οπίσθιου μέρους. Η περιγραφή και το κίνητρο των διεπαφών προγραμματισμού δίνονται στη συνέχεια. Τα σχετικά αποσπάσματα παρουσιάζονται στο **Σχήμα** 2.2 .
Το **API** του πελάτη προσφέρει τις απαραίτητες αρχές για **(c1)** σύνδεση με μια πύλη δικτύου αισθητήρων και καταχώρηση ενός αιτήματος, **(c3)** λήψη των δεδομένων από τους κόμβους αισθητήρων και **(c2)** αποσύνδεση από την πύλη. Για να ενισχυθεί η ευελιξία, ο τύπος των ονομάτων είναι ορισμένος ως αλφαριθμητικά (**strings**) και οι τιμές των δεδομένων λαμβάνονται ως πίνακες σειρές ψηφιολέξεων (**byte arrays**) για να ερμηνευτούν από την εφαρμογή των

```
class NodeInfo {

    byte [] nodeID;
    String itype;
    byte [] value;
}

 // client API

 class SensorNetClient {

 (c1) SensorNetClient (String gwadr,
                       int gwport,
                       String itype,
                       int samplPrd,
                       int ntfyThr);
 (c2) void close();
 (c3) NodeInfo getNxt();
}

 // generic gateway API

 abstract class SensorNetGateway {

 (g1) void start();
 (g2) void stop();
 (g3) void nfyClients(NodeInfo ni,
                      int ntfyThr);
 (g4) void updateNodes(String itype,
                       int smplPrd,
                       int ntfyThr);
}
```

Σχήμα 2.2: Διεπαφές προγραμματισμού SNG

πελατών ανάλογα με τον τύπο που ζητήθηκε. Το **API** της πύλης είναι υπεύθυνο για την αλληλεπίδραση μεταξύ της πύλης και του οπίσθιου μέρους. Περιλαμβάνει τις αρχές για **(g1)** την έναρξη και **(g2)** την παύση της πύλης, **(g3)** την προώθηση των δεδομένων των αισθητήρων πίσω στους πελάτες και **(g4)** την διάδοση των αιτημάτων πελατών στο δίκτυο των αισθητήρων. Η τελευταία μέθοδος είναι αφηρημένη και πρέπει να υλοποιηθεί από το οπίσθιο μέρος. Είναι επίσης σημαντικό να σημειωθεί ότι λόγω της συνάθροισης που εκτελείται από την πύλη, ένα αίτημα πελάτη δεν προκαλεί πάντα μια κλήση της αντίστοιχης μεθόδου πύλης/οπίσθιου μέρους και η πύλη δεν ενημερώνει απαραιτήτως έναν πελάτη κάθε φορά που λαμβάνει πληροφορίες από το οπίσθιο μέρος.

Δεδομένου της ιδιαίτερης φύσης των ασύρματων δικτύων αισθητήρων, αποφασίστηκε να υιοθετηθεί η σημασιολογία (**semantics**) "τελικά τουλάχιστον μια φορά" για την προώθηση των αιτημάτων των πελατών από την πύλη προς τους κόμβους αισθητήρων και η σημασιολογία "το πολύ μια φορά" για τη μετάδοση των δεδομένων των αισθητήρων από τους κόμβους προς την πύλη. Η υιοθετημένη σημασιολογία καθιστά πιθανή την επιτυχή αξιοπιστία σε μια ασύγχρονη μόδα και με άφθονη ευελιξία από άποψη ότι τα πρωτόκολλα που μπορούν να υιοθετηθούν για την υλοποίηση της επικοινωνίας μεταξύ του οπίσθιου μέρους και των κόμβων αισθητήρων.

2.5 Επισκόπηση της υλοποίησης

Το τμήμα της πύλης υλοποιείται ως μια διαδικασία **daemon** που αφουγκράζεται τα αιτήματα των πελατών. Η επικοινωνία μεταξύ του πελάτη και της πύλης επιτυγχάνεται χρησιμοποιώντας ένα απλό πρωτόκολλο στην κορυφή του **TCP/IP**, το οποίο περιγράφεται εν συντομία παρακάτω.

Όταν ένα τμήμα πελάτη αρχικοποιείται, ανοίγει μια σύνδεση με τον **daemon** της πύλης και στέλνει το αίτημα του. Περιμένει έπειτα τα εισερχόμενα δεδομένα των κόμβων που τοποθετούνται σε μια ουρά αναμονής **FIFO**, η οποία μπορεί να επιθεωρηθεί ασύγχρονα από το νήμα της εφαρμογής.

Όταν ο **daemon** της πύλης λαμβάνει ένα αίτημα σύνδεσης πελάτη, "ανοίγει" ένα νήμα (**thread**) για να εξυπηρετήσει τον πελάτη. Το νήμα υπηρεσίας λαμβάνει και καταχωρεί το αίτημα του πελάτη. Εάν οι ζητούμενες ρυθμίσεις ικανοποιούνται από τις αθροισμένες ρυθμίσεις των αιτημάτων που έχουν ήδη εκδοθεί από άλλους πελάτες, δεν γίνεται τίποτα. Αλλιώς, ένα νέο αίτημα που απεικονίζει τις ενημερωμένες αθροισμένες ρυθμίσεις πελατών διανέμεται στο οπίσθιο μέρος.

Η πύλη αποθηκεύει τα δεδομένα που παραλαμβάνονται από τους κόμβους των αισθητήρων, μέσω του οπίσθιου μέρους, σε μια κοινή δομή δεδομένων. Στη συνέχεια, τα νήματα υπηρεσιών (**service threads**) επιθεωρούν περιοδικά τη δομή, προκειμένου να εντοπίσουν και να προωθήσουν τα δεδομένα των αισθητήρων στους πελάτες τους σύμφωνα με τις ζητούμενες ρυθμίσεις. Τα δεδομένα για τα οποία δεν υπάρχει κανένα ενδιαφέρον από πελάτες, θεωρούνται αυτόματα απορρίματα που συλλέγονται για την απελευθέρωση των πόρων μνήμης του μηχανήματος πύλης.

Κεφάλαιο 3

Λογισμικό για υποστήριξη αποφάσεων στην Πυρόσβεση Κλειστών Χώρων

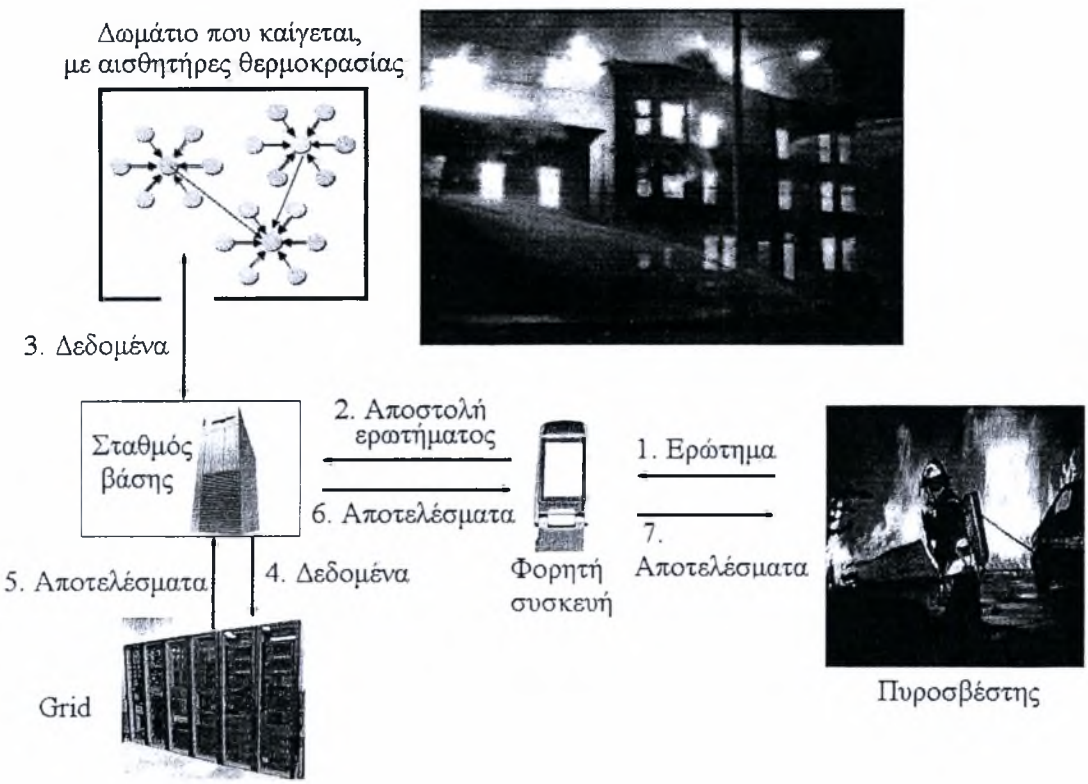
3.1 Περίληψη

Στόχος της παρούσας εργασίας είναι η ανάπτυξη και μοντελοποίηση μιας υποδομής ικανής να ενσωματώσει πολύ ετερογενείς υπολογισμούς, δίκτυα επικοινωνίας και αισθητήρων στα πλαίσια των συστημάτων υποστήριξης απόφασης σε καταστάσεις πυρκαγιάς. Για να επιτύχουμε αυτούς τους στόχους, αναπτύξαμε ένα σύστημα που θα παρέχει προβλέψεις βάσει των δεδομένων των αισθητήρων και της καθοδηγούμενης προσομοίωσης για τον έλεγχο της περιστασιακής αξιολόγησης και την υποστήριξη απόφασης που απαιτείται από το πυροσβεστικό προσωπικό.

3.2 Προδιαγραφές αντιμετώπισης σεναρίου πυρκαγιάς

Παρουσιάζουμε ένα κοινό σενάριο για διάφορες καταστάσεις πυρόσβεσης. Το **Σχήμα** 3.1 απεικονίζει ένα σχηματικό διάγραμμα αυτού του σεναρίου.

Ένα δωμάτιο ή ένα σύνολο από δωμάτια θεωρείται ως ο χώρος, που καθορίζεται από κάποιον όγκο, που μας ενδιαφέρει και όπου μια πυρκαγιά μπορεί να ξεσπάσει. Το δωμάτιο που μας ενδιαφέρει είναι εξοπλισμένο με ένα δίκτυο αισθητήρων. Ακόμα υπάρχει ένας σταθμός βάσης, ο οποίος είναι σε θέση να συλλέγει τα δεδομένα εξόδου των αισθητήρων ή/και του **cluster** δυναμικά. Θεωρούμε αυτά τα δεδομένα, τιμές θερμοκρασίας στις σχετιζόμενες και προκαθορισμένες θέσεις αισθητήρων. Μπορούν επίσης να είναι συντελεστές που αντιστοιχούν στα πολυώνυμα παρεμβολής που αντιπροσωπεύουν τους τομείς κατανομής θερμοκρασίας στην περιοχή του δωματίου. Αυτό υπονοεί ότι οι αισθητήρες μπορούν να επεκταθούν από μη



Σχήμα 3.1: Σενάριο πυρκαγιάς

ευφυή θερμοηλεκτρικά ζεύγη, σε πολύ ευφυείς αισθητήρες με τους σχετικούς ενσωματωμένους μικροεπεξεργαστές για τον τοπικό υπολογισμό των απαραίτητων συντελεστών.

Σήμερα υπάρχουν αισθητήρες που είναι ασύρματα προσβάσιμοι (μέσω **802.11x** ή **Bluetooth**) που είναι σε θέση να τρέχουν **HTTPD servers** στο ενσωματωμένο υλικό έτσι ώστε να είναι προσβάσιμοι μέσω **HTTP**. Ένα σύνολο από υπολογιστικά συστήματα υψηλής απόδοσης (**HPC**) μπορεί να θεωρηθεί ως ο κατάλληλος πόρος του **GRID**. Όταν μια πυρκαγιά ξεσπάσει στο δωμάτιο και ο πυροσβέστης καταφτάσει, τότε χρησιμοποιεί το φορητό του/της (**PDA**) ή το κατάλληλο κινητό τηλέφωνο ή άλλη φορητή συσκευή για να δρομολογήσει ερωτήματα προς το σύστημα για να τον/την βοηθήσει να αξιολογήσει την τρέχουσα κατάσταση (η οποία είναι βασισμένη στις μετρήσεις των αισθητήρων) και το πιο σημαντικό για να ζητήσει απαντήσεις σε ερωτήματα του τύπου:

- Είναι ασφαλές να ανοιχτεί μια συγκεκριμένη πόρτα στα επόμενα 1-2 λεπτά, για να αποκτήσει πρόσβαση στο δωμάτιο ή όχι; ,
- Ποιος ο κίνδυνος και τα επίπεδα εμπιστοσύνης ότι η πυρκαγιά δεν θα διαδοθεί μέσω της πόρτας όταν αυτή ανοιχτεί στα επόμενα 1-2 λεπτά;

Σαφώς τέτοιου είδους ερωτημάτων μπορούν μόνο να απαντηθούν όταν ρυθμίζεται συνεχώς, μια οδηγούμενη από τα δεδομένα των αισθητήρων προσομοίωση της δυναμικής της πυρκαγιάς και οδηγείται από τα ερωτήματα του πυροσβέστη και τα δεδομένα μετρήσεων των αισθητήρων.

3.3 Περιγραφή αλγορίθμου pattern matching

Οι επόμενες συναρτήσεις (**search_pattern**) χρησιμοποιούνται για την εύρεση ακολουθιών συγκεκριμένων θερμοκρασιών και με συγκεκριμένο χρονικό βήμα μεταξύ τους. Πιο συγκεκριμένα, έχουμε δύο συναρτήσεις οι οποίες έχουν ίδια ονόματα, αλλά διαφορετικά ορίσματα. Έτσι ουσιαστικά έχουμε **υπερφόρτωση** (**overloading**) των συναρτήσεων αυτών.

Μια γενική περιγραφή της λειτουργίας των συναρτήσεων αυτών που υλοποιούν τον αλγορίθμο εύρεσης μιας ακολουθίας συγκεκριμένων θερμοκρασιών και με συγκεκριμένο χρονικό βήμα μεταξύ τους είναι η εξής:

Ο αλγόριθμος χωρίζεται σε δύο μέρη: τη δημιουργία των υπό εξέταση "παραθύρων" και τον έλεγχο του κάθε παράθυρου.

Όσον αφορά το πρώτο μέρος, διατρέχουμε τα στοιχεία θερμοκρασίας του ζητούμενου αισθητήρα που υπάρχει στο **vector sensors** μέχρι να βρούμε κάποιο αντικείμενο με θερμοκρασία περίπου ίση (ανάλογα με την τιμή της μεταβλητής **temp_tolerance**) με την πρώτη ζητούμενη θερμοκρασία. Αν βρεθεί κάποιο στοιχείο του αντικειμένου που πληροί τις προυποθέσεις, τότε με τη βοήθεια δύο δεικτών βρίσκουμε που ξεκινά και που τελειώνει το "παράθυρο" που θα εξετάσουμε. Στο παράθυρο αυτό περιέχονται όλα τα στοιχεία (θερμοκρασίας και χρόνου) του αντικειμένου για το χρονικό διάστημα που μπορεί να υφίστανται η ζητούμενη ακολουθία. Δηλαδή, για παράδειγμα αν αναζητούμε την ακολουθία των θερμοκρασιών:

25.0 , 26.0 , 27.0 με χρονικό βήμα **dt = 2 sec** και **time_tolerance** = 0.1

, τότε το κάθε παράθυρο θα έχει διάρκεια ίση με:

(πλήθος αναζητούμενων θερμοκρασιών - 1)***dt** + **time_tolerance** =

= (3 - 1)*2 + 0.1 = 4.1 sec.

Ένα δεύτερο παράδειγμα είναι η αναζήτηση της ακολουθίας των θερμοκρασιών:

25.0, 26.0, 27.0 με χρονικά βήματα **dt = 2, 3 sec** και **time_tolerance** = 0.1

, τότε το κάθε παράθυρο θα έχει διάρκεια ίση με:

(άθροισμα όλων των επιμέρους **dt**) + **time_tolerance** =

= (2 + 3) + **time_tolerance** = 5.1 sec.

Όταν ολοκληρωθεί η κατασκευή του "παραθύρου", τότε η συνάρτηση θα περάσει στο δεύτερο μέρος του αλγορίθμου το οποίο θα εξετάσει το συγκεκριμένο "παράθυρο". Ο αλγόριθμος θα διατρέξει το "παράθυρο" και αν βρει την ζητούμενη ακολουθία θερμοκρασιών (ανάλογα και με τις ζητούμενες ανοχές), τότε αυτή θα αποθηκευτεί σε ένα **vector** με όνομα **result**.

Το δεύτερο μέρος του αλγορίθμου ουσιαστικά σαρώνει το "παράθυρο" και χρησιμοποιεί τις τιμές των παρακάτω αντικειμένων για να επιτύχει το σκοπό του:

- **now_temp, now_time**: είναι οι τρέχουσες τιμές του στοιχείου του αισθητήρα που διατρέχουμε (εντός του "παραθύρου")

- **current_temp, current_time**: είναι οι τρέχουσες ζητούμενες τιμές που έχουμε ήδη βρει

- **next_temp, next_time**: είναι οι τιμές του επόμενου ζητούμενου στοιχείου

Ουσιαστικά, σαρώνουμε το "παράθυρο" και κάθε φορά αναζητούμε τις τιμές θερμοκρασίας και χρόνου που ταιριάζουν στις τιμές των **next_temp** και **next_time** αντίστοιχα. Όταν βρεθεί κάποιο στοιχείο που πληροί αυτές τις προϋποθέσεις, τότε οι τιμές των **current_temp** και **current_time** γίνονται ίσες με τις τιμές των **next_temp, next_time** και οι τελευταίες παίρνουν τις επόμενες αναζητούμενες τιμές σύμφωνα με το **vector T** και το αντίστοιχο χρονικό βήμα **dt**.

Κάθε στοιχείο αποτελέσματος που αποθηκεύεται στο **vector result**, αποθηκεύεται με διαφορετική μορφή, ώστε να είναι η συνάρτηση πιο λειτουργική. Πιο συγκεκριμένα κάθε στοιχείο αποτελέσματος, το οποίο αποτελείται από μία τιμή θερμοκρασίας και μία τιμή χρόνου, αποθηκέυεται ως μία δυάδα διαδοχικών αντικειμένων τύπου **temp** και **time** αντιστοιχα.

Οπότε αν για το συγκεκριμένο "παράθυρο" το **vector result** δεν περιέχει το κατάλληλο πλήθος στοιχείων (το οποίο είναι σύμφωνα με τα παραπάνω ίσο με το διπλάσιο από το πλήθος των αντικειμένων του **T**), αυτό σημαίνει ότι η αναζητούμενη ακολουθία δεν βρέθηκε και τυπώνεται στην οθόνη το σχετικό μήνυμα, ενώ αν ισχύουν οι κατάλληλες προϋποθέσεις, τυπώνει στην οθόνη το σχετικό μήνυμα καθώς και τα στοιχεία του αποτελέσματος.

Η χρήση δύο και όχι μιας συνάρτηση εφαρμόζεται γιατί το ζητούμενο χρονικό βήμα μπορεί να είναι σταθερό ή όχι. Στην πρώτη περίπτωση για την λειτουργία της συνάρτησης **search_pattern**, το χρονικό βήμα υφίσταται ως μία μεταβλητή τύπου **double** ενώ στην δεύτερη περίπτωση για την λειτουργία της συνάρτησης **search_pattern**, το χρονικό βήμα υφίσταται ως ένα **vector**, το οποίο περιέχει αντικείμενα τύπου **time**.

Ας ξεκινήσουμε με την περίπτωση που θέλουμε να βρούμε μία συγκεκριμένη ακολουθία θερμοκρασιών (οποιουδήποτε πλήθους), χρησιμοποιώντας σταθερό χρονικό βήμα μεταξύ τους. Τότε θα καλέσουμε την συνάρτηση:

- **search_pattern(int id, Vector T, double dt, double temp_tolerance, double time_tolerance)**

Ο ακέραιος **id** αποτελεί την "ταυτότητα" του αισθητήρα, του οποίου τα στοιχεία θα ερευνήσουμε. Το **vector T** περιέχει τις θερμοκρασίες που αναζητούμε σε μορφή αντικειμένων τύπου **temp**. Η μεταβλητή **double dt** είναι το σταθερό χρονικό βήμα μεταξύ των θερμοκρασιών. Ακόμα οι τιμές θερμοκρασίας και χρόνου μπορούν να έχουν μία σχετική ανοχή, η οποία καθορίζεται απο τις τιμές των μεταβλητών **temp_tolerance** και **time_tolerance** που δέχεται η συνάρτηση ως όρισματα.

Όσον αφορά τη δεύτερη περίπτωση, όπου θέλουμε να βρούμε μία συγκεκριμένη ακολουθία θερμοκρασιών (οποιουδήποτε πλήθους), χρησιμοποιώντας μεταβλητό χρονικό βήμα μεταξύ τους, θα καλέσουμε την συνάρτηση:

- **search_pattern(int id, Vector T, Vector dt, double temp_tolerance, double time_tolerance)**

Όμοια, ο ακέραιος **id** αποτελεί την "ταυτότητα" του αισθητήρα, του οποίου τα στοιχεία θα ερευνήσουμε, το **vector T** περιέχει τις θερμοκρασίες που αναζητούμε σε μορφή αντικειμένων τύπου **temp** και το **vector dt** περιέχει τα αντικείμενα τύπου **time** που αναπαριστούν τα ενδιάμεσα χρονικά βήματα μεταξύ των θερμοκρασιών. Ακόμα οι τιμές θερμοκρασίας και χρόνου μπορούν να έχουν μία σχετική ανοχή, η οποία καθορίζεται απο τις τιμές των μεταβλητών **temp_tolerance** και **time_tolerance** που δέχεται η συνάρτηση ως όρισματα.

3.4 Επισημάνσεις για την εκτέλεση του λογισμικού πακέτου

Υπάρχουν δύο τρόποι εκκίνησης της λειτουργίας του προγράμματος, ανάλογα με το αν επιθυμούμε την εμφάνιση των βοηθητικών μηνυμάτων σχετικά με την ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων ή όχι. Αν δεν επιθυμούμε την εμφάνιση των βοηθητικών μηνυμάτων, τότε η εκκίνηση του προγράμματος πραγματοποιείται με την εντολή:

java jade.Boot server:ServerAgent compare:CompareAgent gateway:GatewayAgent pda:pdaAgent

Αν επιθυμούμε την εμφάνιση των βοηθητικών μηνυμάτων, τότε η εκκίνηση του προγράμματος πραγματοποιείται με την εντολή:

java jade.Boot -verbosity 1 server:ServerAgent compare:CompareAgent gateway:GatewayAgent pda:pdaAgent

Μερικές ακόμα παρατηρήσεις:

- Θεωρούμε ότι οι ηλεκτρονικοί υπολογιστές, στους οποίους τρέχουν οι πράκτορες **CompareAgent** και **GatewayAgent** αποτελούν σύγχρονες υπολογιστικές μονάδες με αρκετή ισχύ και μνήμη, ώστε να πραγματοποιούνται οι επιθυμητές ενέργειες με ταχύτητα.

- Το δίκτυο των αισθητήρων λειτουργεί ακριβώς με τον ίδιο τρόπο που φαίνεται στην εργασία. Αλλάζει όμως το γεγονός ότι στην περιπτωσή μας οι τιμές που επιστρέφονται, διαβάζονται από το αρχείο **sensor.dat**

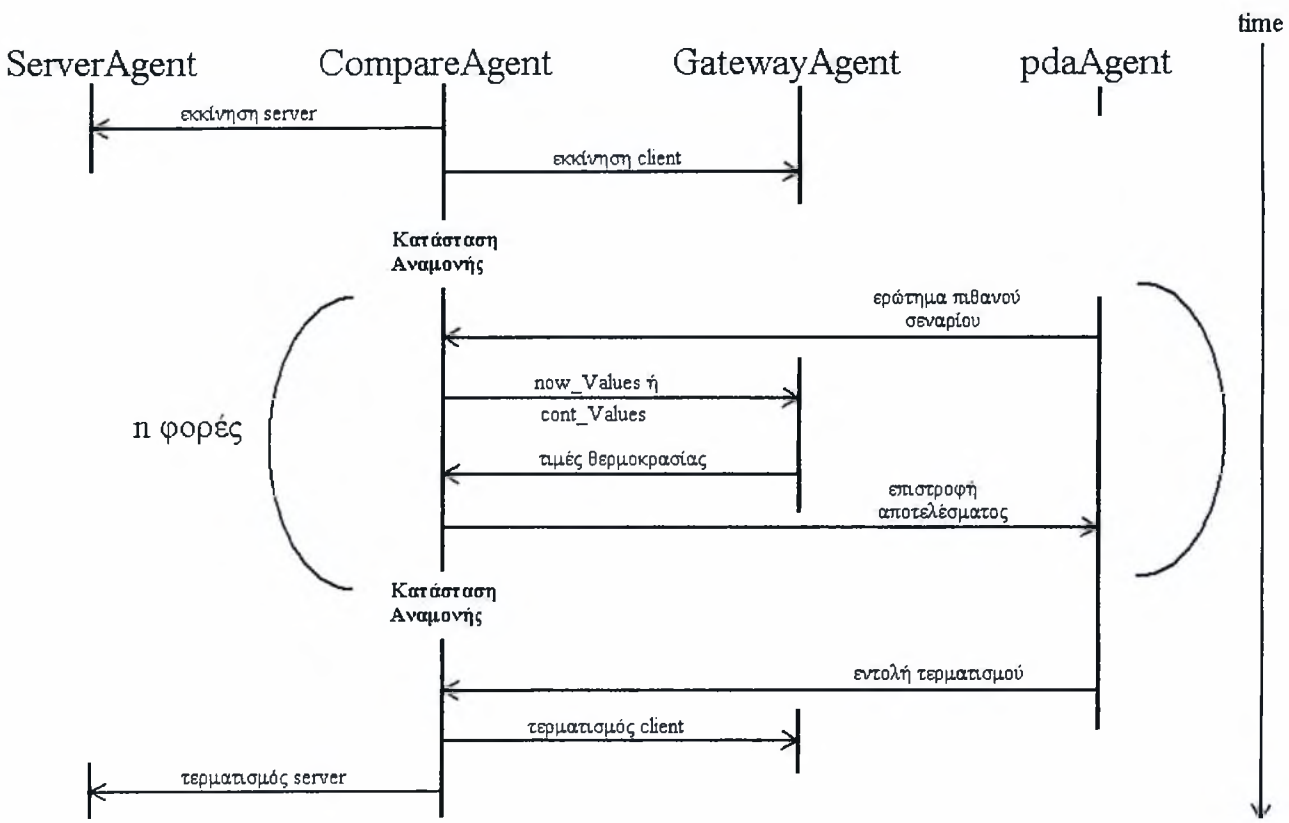
- Το πρόγραμμα λειτουργεί με βάση ένα αρχείο εξόδου του **FDS**. Με κατάλληλες τροποποιήσεις μπορεί να επεκταθεί, ώστε να μπορεί να "διαβάζει" περισσότερα αρχεία.

- Το πόσες τιμές θερμοκρασίας θα επιστρέφει ο πράκτορας **GatewayAgent** στον πράκτορα **CompareAgent** εξαρτάται από την τιμή της μεταβλητής **double dt** του **CompareAgent**. Θεωρούμε ότι κάθε 0.1 **sec** έχουμε μία τιμή θερμοκρασίας για κάθε αισθητήρα, οπότε για παράδειγμα, με τιμή της **dt** ίση 0.2 **sec**, θα μας επιστραφούν 2 τιμές θερμοκρασίας για κάθε αισθητήρα. Μεταβάλλοντας την τιμή της **dt** μεταβάλλουμε το πλήθος των τιμών θερμοκρασίας που μας επιστρέφει ο **GatewayAgent** και με αυτό τον τρόπο αυξάνουμε ή μειώνουμε την πιθανότητα ταυτοποίησης των πιθανών σεναρίων. Η προκαθορισμένη τιμή της μεταβλητής **dt** είναι **0.2 sec** (δηλ. 2 τιμές θερμοκρασίας για κάθε αισθητήρα).

- Η εύρεση των αναζητούμενων θερμοκρασιών στο αρχείο τιμών του **FDS** πραγματοποιείται με τη χρήση τιμών ανοχών (θερμοκρασίας και χρόνου). Οι τιμές αυτές (ανοχές) μπορούν να αλλάξουν ανάλογα με την επιθυμητή ακρίβεια, μεταβάλλοντας τις τιμές των μεταβλητών **double temp_tolerance** και **double time_tolerance** του **CompareAgent**. Οι προκαθορισμένες τιμές των μεταβλητών **temp_tolerance** και **time_tolerance** είναι **0.1** και **0.01** αντίστοιχα.

- Το πλήθος των αισθητήρων (μεταβλητή **no_of_sensors**) "διαβάζεται" αυτόματα από το αρχείο εξόδου του **FDS**.

- Στο **Σχήμα** 3.2 φαίνεται η διαδικασία εύρεσης πιθανού σεναρίου.



Σχήμα 3.2: Διαδικασία Εύρεσης Πιθανού Σεναρίου

- Παρακάτω φαίνεται η εσωτερική μορφή του αρχείου **sensor_data.txt**, μετά από κάποιο πείραμα. Στην αρχή βλέπουμε την ημερομηνία και την ώρα και στη συνέχεια τα δεδομένα από το δίκτυο των αισθητήρων που έχουν καταγραφεί. Σε περίπτωση που ο **client** σταματήσει και επανεκκινήσει, οι "παλιές" τιμές δε διαγράφονται, αλλά παραμένουν στο αρχείο με τη σχετική ημερομηνία και ώρα.

2007-07-20T10:01:40.64

Sensor id: 1

Coord x: 1.50
Coord y: 0.70
Coord z: 2.950

=== Time — Temp ===

=== 0.1 — 9.0 ===
=== 0.2 — 1.0 ===
=== 0.3 — 209.106 ===
=== 0.4 — 211.106 ===
=== 0.5 — 9.0 ===

Sensor id: 2

Coord x: 1.50
Coord y: 3.0
Coord z: 2.95

=== Time — Temp ===

=== 0.1 — 2.0 ===
=== 0.2 — 226.175 ===
=== 0.3 — 229.175 ===

=== 0.4 — 10.0 ===
=== 0.5 — 18.0 ===

Sensor id: 3

Coord x: 3.30
Coord y: 0.70
Coord z: 2.95

=== Time — Temp ===

=== 0.1 — 3.0 ===
=== 0.2 — 200.69 ===
=== 0.3 — 202.69 ===
=== 0.4 — 11.0 ===
=== 0.5 — 13.0 ===

Sensor id: 4

Coord x: 3.30
Coord y: 3.0
Coord z: 2.95

=== Time — Temp ===

=== 0.1 — 4.0 ===
=== 0.2 — 250.298 ===
=== 0.3 — 256.298 ===
=== 0.4 — 12.0 ===
=== 0.5 — 14.0 ===

Sensor id: 5

Coord x: 1.5
Coord y: 0.70
Coord z: 0.0

=== Time — Temp ===

=== 0.1 — 5.0 ===
=== 0.2 — 20.1666 ===
=== 0.3 — 23.1666 ===
=== 0.4 — 13.0 ===
=== 0.5 — 16.0 ===

Sensor id: 6

Coord x: 1.5
Coord y: 3.0
Coord z: 0.0

=== Time — Temp ===

=== 0.1 — 6.0 ===
=== 0.2 — 22.2802 ===
=== 0.3 — 24.2802 ===
=== 0.4 — 6.0 ===
=== 0.5 — 14.0 ===

Sensor id: 7

Coord x: 3.30
Coord y: 0.70
Coord z: 0.0

=== Time — Temp ===

=== 0.1 — 20.6891 ===

=== 0.2 — 29.6891 ===
=== 0.3 — 7.0 ===
=== 0.4 — 7.0 ===
=== 0.5 — 15.0 ===

Sensor id: 8

Coord x: 3.30
Coord y: 3.0
Coord z: 0.0

=== Time — Temp ===

=== 0.1 — 8.0 ===
=== 0.2 — 22.8057 ===
=== 0.3 — 28.8057 ===
=== 0.4 — 8.0 ===
=== 0.5 — 16.0 ===

****

Κεφάλαιο 4

Πειράματα

Στο κεφάλαιο αυτό θα δούμε τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν, προκειμένου να ελέγξουμε τη σωστή λειτουργία του λογισμικού.

Η εύρεση των αποτελεσμάτων έχει ελεγχθεί και λειτουργεί σωστά με τη χρήση τιμων ανοχής της θερμοκρασίας και του χρόνου.

4.1 Κατευθείαν τερματισμός

```
##################################
##          Main Menu           ##
##================================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 2

Agent pda@linux-4ap0:1099/JADE terminating...

pda is asking from compareAgent to terminate...

compare is asking from client to stop.

Stop getting data...

Client terminated...

stop

command :stop

NotificationChannel Stoped

Data for all sensors have been stored!

compare is asking from server to stop.

Agent compare@linux-4ap0:1099/JADE terminating...

Stopping server...

Server terminated succesfully.
```

4.2 Όχι εύρεση κάποιου αποτελέσματος (δίχως retry)

```
##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 1

Asking compare for a possible scenario...

No result found!


##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##

Choice: 2


##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice:
```

4.3 Όχι εύρεση κάποιου αποτελέσματος (με retry)

```
##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 1

Asking compare for a possible scenario...

No result found!


##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##

Choice: 1


Retrying to get a possible scenario...

Asking compare for a possible scenario...

No result found!
```

```
##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##


Choice:
```

4.4 Εύρεση ενός αποτελέσματος (δίχως retry)

```
##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 1

Asking compare for a possible scenario...

*******************************************
File name: roome3_12_tc.csv

Starting time: 75.6028
*******************************************


Play the scenario video? (Y/N)
Choice: Y
```


```
##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##


Choice:
```

4.5 Εύρεση ενός αποτελέσματος (με retry)

```
################################
##          Main Menu         ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice: 1

Asking compare for a possible scenario...

No result found!


##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##

Choice: 1


Retrying to get a possible scenario...

Asking compare for a possible scenario...
```

```
*****************************************
File name: roome3_12_tc.csv

Starting time: 75.6028
*****************************************


Play the scenario video? (Y/N)
Choice: N


##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##

Choice:
```

4.6 Εύρεση περισσότερων από ένα αποτελεσμάτων και έξοδος στο κεντρικό μενού δίχως την εμφάνιση τους

```
##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 1

Asking compare for a possible scenario...


#################################################################
##  More than one, possible scenarios found (2)!               ##
##  What do you want to do?                                    ##
##  (1) Show all possible scenarios                            ##
##  (2) Request more values from network                       ##
##  (3) Exit to main menu                                      ##

Choice: 3
```

```
##################################
##          Main Menu           ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice:
```

4.7 Εύρεση περισσότερων από ένα αποτελεσμάτων και εμφάνιση τους

```
##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 1

Asking compare for a possible scenario...


#################################################################
##  More than one, possible scenarios found (2)!               ##
##  What do you want to do?                                    ##
##  (1) Show all possible scenarios                            ##
##  (2) Request more values from network                       ##
##  (3) Exit to main menu                                      ##

Choice: 1
```

```
*****************************************
File name: roome3_12_tc.csv

Starting time:
time 1: 75.6028
time 2: 1206.01
*****************************************


##################################
##          Main Menu           ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice:
```

4.8 Εύρεση περισσότερων από ένα αποτελεσμάτων (με αρχικό retry) και εμφάνιση τους

```
################################
##          Main Menu         ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice: 1

Asking compare for a possible scenario...

No result found!


##############################
##  (1) Retry               ##

##  (2) Exit to main menu   ##

Choice: 1


Retrying to get a possible scenario...

Asking compare for a possible scenario...
```

```
##################################################################

##  More than one, possible scenarios found (2)!          ##

##  What do you want to do?                               ##

##  (1) Show all possible scenarios                       ##

##  (2) Request more values from network                  ##

##  (3) Exit to main menu                                 ##

Choice: 1


*****************************************
File name: roome3_12_tc.csv

Starting time:
time 1: 75.6028
time 2: 1206.01
*****************************************


###################################
##          Main Menu            ##
##==============================##
##  (1) Get possible scenario    ##
##  (2) Exit                     ##

Choice:
```

4.9 Εύρεση περισσότερων από ένα αποτελεσμάτων, αναζήτηση περισσότερων τιμών, όχι επιτυχία και εμφάνιση προηγούμενων αποτελεσμάτων

```
################################
##          Main Menu         ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice: 1

Asking compare for a possible scenario...


#################################################################
##  More than one, possible scenarios found (2)!               ##
##  What do you want to do?                                    ##
##  (1) Show all possible scenarios                            ##
##  (2) Request more values from network                       ##
##  (3) Exit to main menu                                      ##

Choice: 2

Asking compare for an extended possible scenario...

No result found!
```

```
##################################
##  (1) Retry                   ##
##  (2) Exit to previous menu   ##

Choice: 2


################################################################
##  More than one, possible scenarios found (2)!              ##
##  What do you want to do?                                   ##
##  (1) Show all possible scenarios                           ##
##  (2) Request more values from network                      ##
##  (3) Exit to main menu                                     ##

Choice: 1


*******************************************
File name: roome3_12_tc.csv

Starting time:
time 1: 75.6028
time 2: 1206.01
*******************************************
```

```
################################
##          Main Menu         ##
##============================##
##  (1) Get possible scenario ##
##  (2) Exit                  ##

Choice:
```

4.10 Εύρεση περισσότερων από ένα αποτελεσμάτων, αναζήτηση περισσότερων τιμών, όχι επιτυχία και retry

```
##################################
##          Main Menu           ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice: 1

Asking compare for a possible scenario...


#################################################################
##  More than one, possible scenarios found (2)!               ##
##  What do you want to do?                                    ##
##  (1) Show all possible scenarios                            ##
##  (2) Request more values from network                       ##
##  (3) Exit to main menu                                      ##

Choice: 2

Asking compare for an extended possible scenario...

No result found!
```

```
################################
##  (1) Retry                 ##

##  (2) Exit to previous menu  ##

Choice: 1

Retrying to get a possible scenario...

Asking compare for an extended possible scenario...

No result found!


################################
##  (1) Retry                 ##

##  (2) Exit to previous menu  ##

Choice:
```

4.11 Εύρεση περισσότερων από ένα αποτελεσμάτων, αναζήτηση περισσότερων τιμών, επιτυχία και εμφάνιση αποτελέσματος

```
##################################
##          Main Menu           ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice: 1

Asking compare for a possible scenario...


#################################################################
##  More than one, possible scenarios found (2)!               ##
##  What do you want to do?                                    ##
##  (1) Show all possible scenarios                            ##
##  (2) Request more values from network                       ##
##  (3) Exit to main menu                                      ##

Choice: 2

Asking compare for an extended possible scenario...
```

```
*****************************************
File name: roome3_12_tc.csv

Starting time: 75.6028
*****************************************


Play the scenario video? (Y/N)
Choice: N


##################################
##  (1) Retry                   ##

##  (2) Exit to previous menu   ##

Choice:
```

4.12 Εύρεση περισσότερων από ένα αποτελεσμάτων, αναζήτηση περισσότερων τιμών, επιτυχία με περισσότερα από ένα αποτελέσματα και εμφάνιση τους

```
##################################
##          Main Menu           ##
##==============================##
##  (1) Get possible scenario   ##
##  (2) Exit                    ##

Choice: 1

Asking compare for a possible scenario...


################################################################
##  More than one, possible scenarios found (2)!              ##
##  What do you want to do?                                   ##
##  (1) Show all possible scenarios                           ##
##  (2) Request more values from network                      ##
##  (3) Exit to main menu                                     ##

Choice: 2

Asking compare for an extended possible scenario...
```

```
################################################################

##  More than one, possible scenarios found (2)!          ##

##  What do you want to do?                               ##

##  (1) Show all possible scenarios                       ##

##  (2) Request more values from network                  ##

##  (3) Exit to main menu                                 ##

Choice: 1


*****************************************
File name: roome3_12_tc.csv

Starting time:
time 1: 75.6028
time 2: 1206.01
*****************************************


###################################
##          Main Menu            ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice:
```

4.13 Εύρεση περισσότερων από ένα αποτελεσμάτων, αναζήτηση περισσότερων τιμών, επιτυχία με περισσότερα από ένα αποτελέσματα ξανά αναζήτηση τιμών, ξανά περισσότερα από ένα αποτελέσματα και εμφάνιση τους

```
##################################
##          Main Menu           ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice: 1

Asking compare for a possible scenario...


#################################################################
##  More than one, possible scenarios found (2)!               ##
##  What do you want to do?                                    ##
##  (1) Show all possible scenarios                            ##
##  (2) Request more values from network                       ##
##  (3) Exit to main menu                                      ##

Choice: 2

Asking compare for an extended possible scenario...
```

```
##############################################################

##  More than one, possible scenarios found (2)!            ##

##  What do you want to do?                                 ##

##  (1) Show all possible scenarios                         ##

##  (2) Request more values from network                    ##

##  (3) Exit to main menu                                   ##

Choice: 2

Asking compare for an extended possible scenario...

##############################################################

##  More than one, possible scenarios found (2)!            ##

##  What do you want to do?                                 ##

##  (1) Show all possible scenarios                         ##

##  (2) Request more values from network                    ##

##  (3) Exit to main menu                                   ##

Choice: 1
```

```
*****************************************
File name: roome3_12_tc.csv

Starting time:
time 1: 75.6028
time 2: 1206.01
*****************************************


##################################
##          Main Menu           ##
##============================##
##  (1) Get possible scenario  ##
##  (2) Exit                   ##

Choice:
```

Κεφάλαιο 5

Επίλογος

Αναμφισβήτητα η ανάγκη για δημιουργία ενός λογισμικού πακέτου για υποστήριξη πυρόσβεσης είναι μεγάλη. Μέσα σε αυτή την εργασία, υλοποιήσαμε ένα τέτοιο πρότυπο πακέτο στο αρχικό στάδιο που επιδέχεται βελτίωσης και επέκτασης. Αντιμετωπίσαμε τόσο προβλήματα τεχνικά όπως υλικού/λογισμικού, επικοινωνίας δικτύου αισθητήρων με το σύστημα μας ή εμφάνισης του σεναρίου στην οθόνη, όσο και λογικά όπως "πως θα επιλεγεί το κατάλληλο scenario". Η κατάλληλη επιλογή πλατφόρμας πρακτόρων (**JADE**) επιλύει τα πρώτα. Ενώ το κατάλληλο λογισμικό προσομοίωσης της φωτιάς (**FDS**) και ένας αλγόριθμος "**pattern matching**" με κάποια ανοχή βοηθά στην αντιμετώπιση των δεύτερων προβλημάτων.

Το λογισμικό που υλοποιείται μπορεί να επεκταθεί με διάφορους τρόπους όσον αφορά την απόδοση, αλλά και την εμφάνιση (**API**). Ορισμένοι από αυτούς περιγράφονται παρακάτω:

- Υλοποίηση ασύρματης επικοινωνίας μεταξύ **pda** και ηλεκτρονικού υπολογιστή μέσω **Bluetooth**
- Χρήση πολλαπλών αρχείων εξόδου του **FDS**
- Δημιουργία γραφικού περιβάλλοντος στο **pda**
- Επέκταση του μενού λειτουργιών του **pda**, ώστε να εκτελεί περισσότερα ή πιο εξειδικευμένα ερωτήματα
- Υλοποίηση περισσότερο αποδοτικού αλγορίθμου για **pattern matching**

Παράρτημα Α

Σχολιασμός, επεξήγηση κώδικα

Α.1 Αισθητήρες και διαχείρισή τους

Α.1.1 Αρχείο temp.java

Η κλάση αυτή παράγει αντικείμενα τύπου **temp**. Τα αντικείμενα αυτά ουσιαστικά αναπαριστούν την κάθε θερμοκρασία και κάθε τέτοιο αντικείμενο περιέχει μόνο την μεταβλητή **double temp**. Οι συναρτήσεις που περιέχονται σε αυτή την κλάση είναι οι εξής:

- **temp(double temp)**: πρόκειται για τον constructor, ο οποίος δημιουργεί αντικείμενα τύπου **temp** με όρισμα την τιμή της θερμοκρασίας
- **get_temp()**: επιστρέφει την τιμή της μεταβλητής **temp**

Α.1.2 Αρχείο time.java

Η κλάση αυτή παράγει αντικείμενα τύπου **time**. Τα αντικείμενα αυτά ουσιαστικά αναπαριστούν την κάθε χρονική στιγμή και κάθε τέτοιο αντικείμενο περιέχει μόνο την μεταβλητή **double time**. Οι συναρτήσεις που περιέχονται σε αυτή την κλάση είναι οι εξής:

- **time(double time)**: πρόκειται για τον constructor, ο οποίος δημιουργεί αντικείμενα τύπου **time** με όρισμα την τιμή της χρονικής στιγμής

- **get_time()**: επιστρέφει την τιμή της μεταβλητής **time**

A.1.3 Αρχείο sensor.java

Η κλάση αυτή παράγει αντικείμενα τύπου **sensor**. Κάθε τέτοιο αντικείμενο περιέχει τις εξής μεταβλητές:

- **int sensor_id**: μοναδικός αριθμός κάθε αισθητήρα (για την αναγνώρισή του)
- **double coord_x**: οι μεταβλητές x, y, z περιγράφουν τη θέση ενός αντικειμένου sensor στο χώρο
- **double coord_y**
- **double coord_z**
- **double [ ] temp**: οι θερμοκρασίες που μετρά ο αισθητήρας
- **double [ ] time**: οι χρονικές στιγμές που ο αισθητήρας μετράει τις θερμοκρασίες

Οι συναρτήσεις που περιέχονται στην κλάση είναι οι εξής:

- **sensor(int sensor_id, double coord_x, double coord_y, double coord_z, double [ ] temp, double [ ] time)**: δεν είναι τίποτα άλλο από τον **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **sensor** και δέχεται ως ορίσματα τον ακέραιο - ταυτότητα του αισθητήρα, τις συντεταγμένες του, τις θερμοκρασίες που μετράει, καθώς και τις χρονικές στιγμές των μετρήσεων.
- **sensor(int sensor_id, double [ ] temp, double [ ] time)**: ο δεύτερος **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **sensor** και δέχεται ως ορίσματα τον ακέραιο - ταυτότητα του αισθητήρα, τις θερμοκρασίες που μετράει, καθώς και τις χρονικές στιγμές των μετρήσεων. Τα πεδία των μεταβλητών που αναπαριστούν τις συντεταγμένες θέτονται μηδέν.
- **get_id()**: επιστρέφει την τιμή της μεταβλητής **sensor_id** του αισθητήρα
- **get_temp()**: επιστρέφει τον πίνακα των θερμοκρασιών **temp** του αισθητήρα

- **get_time()**: επιστρέφει τον πίνακα των χρονικών στιγμών των μετρήσεων **time** του αισθητήρα
- **get_coord_x()**: επιστρέφει την τιμή της μεταβλητής **coord_x** του αισθητήρα
- **get_coord_y()**: επιστρέφει την τιμή της μεταβλητής **coord_y** του αισθητήρα
- **get_coord_z()**: επιστρέφει την τιμή της μεταβλητής **coord_z** του αισθητήρα
- **get_length()**: επιστρέφει το μήκος (μέγεθος) του πίνακα **temp** (το οποίο είναι ίσο με το μέγεθος του πίνακα **time**)
- **get_last_temp()**: επιστρέφει την τιμή της τελευταίας τιμής θερμοκρασίας που υπάρχει στον πίνακα **temp**
- **get_last_time()**: επιστρέφει την τιμή της τελευταίας τιμής χρόνου που υπάρχει στον πίνακα **time**
- **set_sensor_values(int position, double temp, double time, int sensor_id)**: θέτει τις τιμές των πινάκων **temp**, **time** καθώς και της μεταβλητής **sensor_id**
- **has_values()**: αν ο αισθητήρας έχει μετρήσει τουλάχιστον μία τιμή θερμοκρασίας, τότε επιστρέφει **true**, αλλιώς επιστρέφει **false**

A.1.4 Αρχείο sensor_records.java

Η κλάση αυτή διαχειρίζεται τα αντικείμενα τύπου **sensor**. Δημιουργεί ένα **vector** με το όνομα **sensors** και πραγματοποιεί λειτουργίες μέσω των παρακάτω συναρτήσεων:

- **sensor_records()**: πρόκειται για τον **constructor**, ο οποίος αρχικοποιεί το **vector**
- **sensor_records(int size)**: πρόκειται για τον δεύτερο **constructor**, ο οποίος αρχικοποιεί το **vector** με βάση την απαιτούμενη χωρητικότητα
- **add_sensor(sensor s)**: προσθήκη ενός αντικειμένου τύπου **sensor** (s) στο **vector**
- **erase_sensor(sensor s)**: απομάκρυνση ενός αντικειμένου τύπου **sensor** (s) από το **vector**
- **get_nofsensors()**: επιστρέφει το πλήθος των αντικειμένων του **vector**

- **get_sensor_vector()**: επιστρέφει το **vector** των **sensors**
- **search_sensor(int s)**: δέχεται ως όρισμα έναν ακέραιο **s** και ψάχνει στο **Vector sensors** αν υπάρχει κάποιο αντικείμενο τύπου **sensor** με **sensor id** ίσο με **s**. Αν βρεθεί το ζητούμενο αντικείμενο, τότε απομακρύνεται από το **Vector sensors** και επιστρέφεται. Σε αντίθετη περίπτωση επιστρέφει **null**.
- **show_sensors()**: τυπώνει στην οθόνη τα στοιχεία κάθε αισθητήρα που υπάρχει στο **vector**. Αν το **vector** ειναι άδειο, τότε τυπώνει το σχετικό μήνυμα.
- **show_sensor(sensor s)**: δέχεται ως όρισμα ένα αντικείμενο τύπου **sensor** και τυπώνει τα στοιχεία του
- **remove_gaps(String str)**: δέχεται ως όρισμα ένα **String** και απομακρύνει τυχόν κενά που περιέχονται σε αυτό. Τέλος, επιστρέφει το νέο **String**.
- **is_almost(double b, double s, double tolerance)**: η συνάρτηση αυτή χρησιμοποιείται για να συγκρίνουμε αν δύο τιμές είναι περίπου ίσες με μια ανοχή που αντιπροσωπεύεται απο την μεταβλητή **tolerance**. Ουσιαστικά δέχεται ως όρισμα τρεις μεταβλητές τύπου **double** και συγκρίνει τις τιμές των δύο πρώτων. Αν αυτές διαφέρουν κατά τιμή μικρότερη της ανοχής **tolerance**, τότε επιστρέφεται **true**, αλλιώς επιστρέφεται **false**.
- **show_vector(Vector v)**: η συνάρτηση αυτή χρησιμοποιείται για να τυπώσουμε στην οθόνη τα στοιχεία των αντικειμένων τύπου **sensor** που περιέχονται στο **vector v** που παίρνει ως όρισμα.
- **load(File file)**: η συνάρτηση αυτή δέχεται ως όρισμα ένα αντικείμενο τύπου **File** (το οποίο ουσιαστικά είναι ένα αρχείο και πιο συγκεκριμένα στην περίπτωση μας είναι της μορφής *.csv, δηλαδή ένα αρχείο αποτελεσμάτων του **FDS**), το οποίο προσπαθεί να ανοίξει. Αν τα καταφέρει (δηλαδή δεν υπάρξει πρόβλημα λόγω κάποιου **exception**), με κατάλληλη επεξεργασία βρίσκει το πλήθος των αισθητήρων των οποίων οι μετρήσεις υπάρχουν στο αρχείο, καθώς και το πλήθος των "χρήσιμων" γραμμών του αρχείου, δηλαδή εκείνων των γραμμών που περιέχουν μόνο στοιχεία μετρήσεων. Στη συνέχεια, καλεί τη συνάρτηση **vector_load** με κατάλληλα ορίσματα. Τέλος, επιστρέφει το πλήθος των αισθητήρων.

- **vector_load(File file, int no_of_sensors, int no_of_lines)**: η συνάρτηση αυτή καλείται για να "διαβάσει" το αρχείο που αναφέρθηκε παραπάνω και να αποθηκεύσει τα στοιχεία του, τα οποία έχει προηγουμένως μετατρέψει σε αντικείμενα τύπου **sensor**, στο **vector loaded** και αφού ολοκληρωθεί η διαδικασία αυτή, η τιμή του **vector loaded** επιστρέφεται για να αποθηκευτεί στο **vector sensors**.

- **search_pattern(int id, Vector T, double dt, double temp_tolerance, double time_tolerance)**: η συνάρτηση αυτή υλοποιεί τον αλγόριθμο **pattern matching**, όπως αυτός περιγράφεται στο 3ο κεφάλαιο για την περίπτωση που το **dt** είναι μεταβλητή τύπου **double**.

- **search_pattern(int id, Vector T, Vector dt, double temp_tolerance, double time_tolerance)**: η συνάρτηση αυτή υλοποιεί τον αλγόριθμο **pattern matching**, όπως αυτός περιγράφεται στο 3ο κεφάλαιο για την περίπτωση που το **dt** είναι μεταβλητή τύπου **Vector**.

A.2 Πράκτορες και λειτουργίες τους

Η λειτουργία των παραπάνω αρχείων ολοκληρώνεται με την χρήση πρακτόρων μέσω του **Jade**. Για να ανταλλάσσουν οι πράκτορες μεταξύ τους μηνύματα που περιέχουν δεδομένα απαιτείται η δημιουργία νέων αντικειμένων με συγκεκριμενα χαρακτηριστικά (εναλλακτικά απαιτειται η δημιουργία μιας νέας **ontology**). Γι'αυτό δημιουργούμε τα αντικείμενα **querry_object** και **reply_object**. Πιο αναλυτικά, έχουμε τα παρακάτω αρχεία:

A.2.1 Αρχείο requestValues_object.java

Τα αντικείμενα τύπου **requestValues_object** χρησιμοποιούνται για την αποστολή μηνυμάτων ερώτησης μεταξύ δύο πρακτόρων. Ουσιαστικά πρόκειται για τα στοιχεία που απαιτούνται για να πραγματοποιηθεί η ζητούμενη εύρεση ακολουθιών, μέσω των συναρτήσεων **search_pattern** και περιέχει τις εξής μεταβλητές:

- **private String querry**: αυτό το **String** παίρνει τιμές ανάλογα με το είδος του ερωτήματος που επιθυμούμε να πραγματοποιηθεί
- **private double dt**: πρόκειται για μία μεταβλητή **double** που χρησιμοποιείται για να καθορίσουμε το χρονικό διάστημα που μας ενδιαφέρει
- **private int no_of_sensors**: το πλήθος των αισθητήρων

Οι συναρτήσεις που περιέχονται στην κλάση είναι οι εξής:

- **requestValues_object(String querry)**: πρόκειται για τον πρώτο **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **requestValues_object** και δέχεται ως όρισμα ένα **String**, το οποίο δηλώνει τον τύπο του ερωτήματος.
- **requestValues_object(String querry, double dt)**: πρόκειται για τον δεύτερο **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **requestValues_object** και δέχεται ως ορίσματα ένα **String**, το οποίο δηλώνει τον τύπο του ερωτήματος και μία μεταβλητή **double** που αντιπροσωπεύει κάποιο χρονικό διάστημα.
- **requestValues_object(String querry, int no_of_sensors)**: πρόκειται για τον τρίτο **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **requestValues_object**

και δέχεται ως ορίσματα ένα **String**, το οποίο δηλώνει τον τύπο του ερωτήματος και μία μεταβλητή **int** που αντιπροσωπεύει το πλήθος των αισθητήρων.

- **requestValues_object(String querry, double dt, int no_of_sensors)** : πρόκειται για τον τέταρτο **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **requestValues_object** και δέχεται ως ορίσματα ένα **String**, το οποίο δηλώνει τον τύπο του ερωτήματος, μία μεταβλητή **double** που αντιπροσωπεύει κάποιο χρονικό διάστημα και μία μεταβλητή **int** που αντιπροσωπεύει το πλήθος των αισθητήρων.
- **set_querry(String querry)**: θέτει την τιμή της μεταβλητής **querry**
- **set_dt(double dt)**: θέτει την τιμή της μεταβλητής **dt**
- **set_no_of_sensors(int no_of_sensors)**: θέτει την τιμή της μεταβλητής **no_of_sensors**
- **get_querry()**: επιστρέφει την τιμή της μεταβλητής **querry**
- **get_dt()**: επιστρέφει την τιμή της μεταβλητής **dt**
- **get_no_of_sensors()**: επιστρέφει την τιμή της μεταβλητής **get_no_of_sensors**

A.2.2 Αρχείο reply_Values_object.java

Τα αντικείμενα τύπου **reply_Values_object** χρησιμοποιούνται για την αποστολή μηνυμάτων απάντησης (δηλαδή αποτελεσμάτων) ενός πράκτορα προς εκείνο τον πράκτορα που έκανε το ερώτημα. Ουσιαστικά πρόκειται για το αποτέλεσμα της ζητούμενης ενέργειας και περιέχει μόνο την μεταβλητή **private Vector results.**
Οι συναρτήσεις που περιέχονται στην κλάση είναι οι εξής:

- **reply_Values_object(Vector results)**: πρόκειται για τον **constructor** της κλάσης, όπου κατασκευάζει τα αντικείμενα τύπου **reply_Values_object** και δέχεται ως ορίσμα το **Vector** το οποίο περιέχει τα ζητούμενα δεδομένα.
- **set_results(Vector results)**: θέτει την τιμή του **Vector** των αποτελεσμάτων
- **get_results()**: επιστρέφει την τιμή του **Vector** των αποτελεσμάτων

Για να ειναι δυνατή η αποστολή μηνυμάτων που εμπεριέχουν δεδομένα, απαιτείται και τα δύο είδη αντικειμένων (**requestValues_object, reply_Values_object**) να είναι **Serialized**, γι'αυτό και στις δύο κλάσεις προσθέτουμε την δήλωση: **"implements java.io.Serializable"**.
Σύμφωνα με τα παραπάνω, αν θέλει καποιος πράκτορας να στείλει ένα ερώτημα σχετικά με τις τρέχουσες τιμές θερμοκρασιας - χρόνου ή με τις τιμές θερμοκρασιας - χρόνου κατά ένα χρονικό διάστημα (το οποίο αντιπροσωπεύεται από την μεταβλητή **dt**), τότε θα πρέπει να δημιουργήσει ενα αντικείμενο τύπου **requestValues_object** με τα καταλληλα χαρακτηριστικά (δηλαδή στην πρώτη περίπτωση το **String querry** να έχει την τιμη "GiveNowValues", ενώ στην δεύτερη περίπτωση το **String querry** να έχει την τιμη "GiveContValues" καθώς και η μεταβλητή **dt** την ζητούμενη τιμή), να το προσθέσει σε ένα μήνυμα τύπου **ACLMessage** και να το στείλει στον κατάλληλο πράκτορα - παραλήπτη (στην περίπτωση μας τον **GatewayAgent**), ώστε να το επεξεργαστεί. 'Οταν ο πράκτορας - παραλήπτης το λάβει και το "αποκωδικοποιήσει", θα καλέσει την αντίστοιχη συμπεριφορά που υλοποιεί την ενέργεια που θέλουμε να πραγματοποιήσουμε.
Στη συνέχεια, ο πράκτορας **GatewayAgent** θα δημιουργήσει με τη σειρά του ένα αντικείμενο τύπου **reply_Values_object**, το οποίο θα εμπεριέχει το αποτέλεσμα της ζητούμενης ενέργειας και θα το στείλέι με όμοιο τρόπο στον πράκτορα που το ζήτησε. Τέλος ο πράκτορας που τώρα έλαβε το αποτέλεσμα, θα τυπώσει στην οθόνη τα αποτελέσματα χρησιμοποιώντας τη συνάρτηση **show_now_temps(Vector result_values)** (ή στη δεύτερη περίπτωση τη συνάρτηση **show_cont_temps(Vector result_values)**.
Οι τιμές θερμοκρασίας του δικτύου αισθητήρων δίνονται μέσω μιας **TCP** σύνδεσης από κάποιον συγκεκριμένο **server**. Η σύνδεση αυτή υλοποιείται από το μοντέλο **server/client** και για την πραγματοποίηση της απαιτούνται οι εξής ενέργειες:

- **εκκίνηση server**: ο **server** ξεκινά τη λειτουργία του, δηλαδή συνδέεται σε κάποιο συγκεκριμένο (προκαθορισμένο) **port** του συστήματος στο οποίο βρίσκεται και περιμένει αιτήσεις για εξυπηρέτηση.
- **εκκίνηση client**: ο **client** ξεκινά τη λειτουργία του, δηλαδή συνδέεται σε κάποιο συγκεκριμένο (προκαθορισμένο) **port** του συστήματος στο οποίο έχει ήδη συνδεθεί ο **server**.
- **χρήση σύνδεσης**: για να ξεκινήσει ο **client** να δέχεται τιμές απο το δίκτυο, πρέπει να στείλει στον **server** το μήνυμα **"get"**.

- **τερματισμός σύνδεσης**: για να σταματήσει ο **server** να στέλνει τιμές, πρέπει ο **client** να του στείλει το μήνυμα **"stop"**.

Στην περίπτωση μας το ρόλο του **server** αναλαμβάνει να παίξει ένας ακόμη πράκτορας με όνομα **ServerAgent**. Ακόμα ο ρόλος του **client** αντιπροσωπεύεται από τον πράκτορα **GatewayAgent**. Οι ενέργειες και των δύο περιγράφονται παρακάτω.

A.2.3 Αρχείο ServerAgent.java

Ο πράκτορας **ServerAgent** υλοποιεί την λειτουργία του **server**. Κατά την εκκίνηση του ξεκινά την εκτέλεση της συμπεριφοράς **st**, η οποία διαβάζει τα εισερχόμενα μηνύματα και καλεί ή σταματά την συμπεριφορά που είναι υπεύθυνη για την αποστολή των τιμών στον **client**.
Αφού του δοθεί εντολή για εκκίνηση, ο πράκτορας αναμένει για κάποιο μήνυμα τύπου **ACLMessage**. 'Οταν δεχτεί ένα τέτοιο μήνυμα, διαβάζονται τα περιεχόμενα του. Το μήνυμα θα περιέχει ένα αντικείμενο τύπου **requestValues_object**, το οποίο ουσιαστικά μας μεταφέρει το αίτημα, μέσω του **String querry** που περιέχει. Κατόπιν θα διαβαστεί το **String querry** και ανάλογα με την τιμή του, ο πράκτορας θα εκτέλεσει μία απο τις παρακάτω ενέργειες:

- Αν το **String querry** έχει τιμή **"StartServer"**: τότε έχουμε εκκίνηση του **server** στο **port 4444** (μπορούμε να χρησιμοποιήσουμε οποιοδήποτε ελεύθερο **port**, συνήθως μεγαλύτερο από το 1063, αρκεί στο ίδιο **port** να συνδεθεί και ο **client**). Εφόσον πραγματοποιηθούν όλες οι απαραίτητες ενέργειες επιτυχώς (υπάρχει περίπτωση να εμφανιστεί κάποιο **Exception**, λόγω σφάλματος χρήστη ή συστήματος), ο **server** είναι έτοιμος και αναμένει αιτήσεις για εξυπηρέτηση.
  (εκτέλεση συμπεριφοράς **b** σε **dedicated thread**)
- Αν το **String querry** έχει τιμή **"StopServer"**: τότε έχουμε τερματισμό του **server**, δηλαδή κλείσιμο του **port 4444**.
  (διακοπή συμπεριφοράς **b**)

Για την λειτουργία του **server** χρησιμοποιούνται εντολές που περιέχονται στα αρχεία **Dispatcher** και **NotificationChannel**, οι οποίες αφορούν την λεπτομερή υλοποίηση του μοντέλου και σκόπιμα δεν σχολιάζονται για λόγους απλότητας.

A.2.4 Αρχείο Client.java

Για απλοποίηση μέρους του κώδικα, υλοποιούμε το μοντέλο του **client** ως ένα αντικείμενο με όνομα **Client**. Περιληπτικά δημιουργούμε ένα αντικείμενο τύπου **Client** χρησιμοποιώντας τον **constructor** της κλάσης με ορίσματα το όνομα του **host** και τον αριθμό του **port**:

- **Client(String host, int port)**

Όταν τελειώσουμε με την χρήση του αντικειμένου, χρησιμοποιούμε τη συνάρτηση:

- **close()**

η οποία κάνει τις απαιτούμενες ενέργειες τερματισμού (κλείνει το **socket**, κλπ.).

A.2.5 Αρχείο MyComparator.java

Πρόκειται για μία κλάση, η οποία χρησιμοποιείται από τον πράκτορα **GatewayAgent** για την ταξινόμηση ενός **Vector** με αντικείμενα τύπου **sensor**, ως προς τη μεταβλητή **sensor_id**.

A.2.6 Αρχείο GatewayAgent.java

Ο πράκτορας **GatewayAgent** εκτελεί χρέη εξυπηρέτη. Κατά την εκκίνηση του ξεκινά την εκτέλεση της συμπεριφοράς **st**, η οποία διαβάζει τα εισερχόμενα μηνύματα και καλεί τις αντίστοιχες συμπεριφορές για τη συλλογή των ζητούμενων δεδομένων και την αποστολή μηνυμάτων απάντησης.

Δέχεται ερωτήματα με τη μορφή μηνυμάτων τύπου **ACLMessage**, τα οποία περιέχουν ένα αντικείμενο τύπου **requestValues_object** και ανάλογα με το περιεχόμενο του, εκτελεί τις παρακάτω ενέργειες:

- **περιεχόμενο**: **String** με τιμή **"StartClient"**: εκκίνηση λειτουργίας **client**, δηλαδή απόπειρα σύνδεσης με τον **server**, εκκίνηση λήψης τιμών θερμοκρασίας από το δίκτυο και ταυτόχρονη αποθήκευση των τιμών σε κατάλληλη δομή δεδομένων (**sensor_records**). (εκτέλεση συμπεριφοράς **b**)

- **περιεχόμενο**: **String** με τιμή **"StopClient"**: τερματισμός λειτουργίας **client**, δηλαδή απόπειρα αποσύνδεσης από τον **server**, παύση λήψης τιμών θερμοκρασίας από το δίκτυο και παύση αποθήκευσης των τιμών αυτών. Μόλις σταματήσει η αποθήκευση των τιμών, καλείται η συνάρτηση **save_current_data(sensor_records s)**, η οποία αποθηκεύει τα μέχρι τώρα ληφθέντα δεδομένα σε ένα αρχείο με όνομα **sensor_data.txt**, μαζί με την τρέχουσα ημερομηνία και ώρα, αφού πρώτα ταξινομήσει τα δεδομένα αυτά με βάση το **sensor_id**. Η ταξινόμηση αυτή πραγματοποιείται με τη χρήση της συνάρτησης **sort_vector(Vector s)**.
  (διακοπή συμπεριφοράς **b**)

- **περιεχόμενο**: **String** με τιμή **"GiveNowValues"**: συλλογή των τρέχουσων τιμών θερμοκρασίας - χρόνου (δηλαδή την τελευταίες τιμές που έχουμε συλλέξει από το δίκτυο), αποθήκευση του αποτελέσματος και ταξινόμηση των τιμών του μέσω της συνάρτησης **sort_vector(Vector s)**, αποθήκευσή του σε ένα αντικείμενο τύπου **reply_Values_object** και αποστολή του αποτελέσματος στον πράκτορα που έστειλε το ερώτημα.
  (εκτέλεση συμπεριφοράς **c**)

- **περιεχόμενο**: **String** με τιμή **"GiveContValues"** και **double dt** με κάποια τιμή: συλλογή των τιμών θερμοκρασίας - χρόνου για χρονικό διάστημα ίσο με την τιμή της μεταβλητής **dt**, αποθήκευση του αποτελέσματος και ταξινόμηση των τιμών του μέσω της συνάρτησης **sort_vector(Vector s)**, αποθήκευσή του σε ένα αντικείμενο τύπου **reply_Values_object** και αποστολή του αποτελέσματος στον πράκτορα που έστειλε το ερώτημα.
  (εκτέλεση συμπεριφοράς **d**)

Για την διασφάλιση της παράλληλης λειτουργίας όλων των διαθέσιμων συμπεριφορών του πράκτορα **GatewayAgent**, κάθε συμπεριφορά που καλείται, εκτελείται σε ένα δικό της νήμα (**dedicated thread**). Αυτό καθίσταται δυνατό μέσω της χρήσης του αντικειμένου **ThreadedBehaviourFactory** που μας παρέχει το **Jade**.

A.2.7 Αρχείο pda_reply_object.java

Τα αντικείμενα τύπου **pda_reply_object** χρησιμοποιούνται για την αποστολή μηνυμάτων απάντησης προς τον πράκτορα του **pda** (**pdaAgent**) και κάθε ένα περιέχει δύο **Vector**: το **Vector**

file_names που περιέχει τα ονόματα των αρχείων στα οποία περιλαμβάνονται τα αποτελέσματα και το **Vector start_time**, στο οποίο περιλαμβάνονται οι χρόνοι των αποτελεσμάτων.

A.2.8 Αρχείο CompareAgent.java

Πρόκειται για έναν πράκτορα, ο οποίος ουσιαστικά ευθύνεται για την οργάνωση της λειτουργίας του όλου λογισμικού. Κατά την εκκίνησή του, αρχικά δίνει εντολή στους κατάλληλους πράκτορες για να εκκινήσουν την λειτουργία τους ως **server/client**, ώστε να συλλεχθούν και αποθηκευτούν οι απαιτούμενες τιμές θερμοκρασίας που έχουν καταγραφεί από το δίκτυο των αισθητήρων. Κατόπιν, "φορτώνεται" το κατάλληλο αρχείο **.csv** (που περιέχει τα δεδομένα προσομοίωσης του **FDS**) και καλείται η συμπεριφορά **b** να εκκινήσει, η οποία αναμένει για μηνύματα τύπου **ACLMessage**. Όταν κάποιο τέτοιο μήνυμα καταφτάσει, τότε διαβάζονται τα περιεχόμενα του (λογικά θα περιέχεται ένα αντικείμενο τύπου **requestValues_object** το οποίο μεταφέρει το ερώτημα) και ανάλογα με την τιμή του **String querry**, πραγματοποιούνται οι εξής ενέργειες:

- **περιεχόμενο**: **String** με τιμή **"getScenario"**: έχουμε εκτέλεση της συμπεριφοράς **e**, η οποία αρχικά ζητά από τον πράκτορα **GatewayAgent** τις τρέχουσες τιμές θερμοκρασίας που επικρατούν στο χώρο που μας ενδιαφέρει, μέσω της συνάρτησης **get_NowValues** και στη συνέχεια καλεί τη συνάρτηση
  search_patt(no_of_sensors, nowValues, temp_tolerance, time_tolerance) με ορίσματα το πλήθος των αισθητήρων, τις τρέχουσες θερμοκρασίες και τις ανοχές θερμοκρασίας και χρόνου. Αυτή με τη σειρά της καλεί τη συνάρτηση **search_pattern** για κάθε αισθητήρα και αποθηκεύει τα επιμέρους αποτελέσματα χρόνων που επιστρέφονται. Αφού τελειώσει η ενέργεια αυτή και αν έχουν βρεθεί χρόνοι αποτελεσμάτων για όλους τους αισθητήρες (δηλαδή οι τρέχουσες θερμοκρασίες για κάθε αισθητήρα υπάρχουν μέσα στο αρχείο του **FDS**), τότε ελέγχονται όλα τα επιμέρους αποτελέσματα χρόνων για κοινές τιμές, μέσω των συναρτήσεων **exists_in** και **is_almost**. Κάθε χρόνος που ταυτίζεται σε όλα τα επιμέρους αποτελέσματα ορίζεται σαν αποτέλεσμα και αποθηκεύεται σε κατάλληλη δομή. Τέλος, το αποτέλεσμα αποθηκεύεται σε ένα αντικείμενο τύπου **pda_reply_object** και αποστέλλεται στον **pda** πράκτορα, που έκανε το ερώτημα.
- **περιεχόμενο**: **String** με τιμή **"getExtendedScenario"**: έχουμε εκτέλεση της

συμπεριφοράς **f**, η οποία αρχικά ζητά από τον πράκτορα **GatewayAgent** τις τιμές θερμοκρασίας, που επικρατούν στο χώρο που μας ενδιαφέρει, για χρονικό διάστημα **dt** μέσω της συνάρτησης **get_ContValues** και στη συνέχεια καλεί τη συνάρτηση **search_patt(no_of_sensors, contValues, temp_tolerance, time_tolerance)** με ορίσματα το πλήθος των αισθητήρων, τις τιμές θερμοκρασίας για χρονικό διάστημα **dt** και τις ανοχές θερμοκρασίας και χρόνου. Αυτή με τη σειρά της καλεί τη συνάρτηση **search_pattern** για κάθε αισθητήρα και αποθηκεύει τα επιμέρους αποτελέσματα χρόνων που επιστρέφονται. Αφού τελειώσει η ενέργεια αυτή και αν έχουν βρεθεί χρόνοι αποτελεσμάτων για όλους τους αισθητήρες (δηλαδή οι τρέχουσες θερμοκρασίες για κάθε αισθητήρα υπάρχουν μέσα στο αρχείο του **FDS**), τότε ελέγχονται όλα τα επιμέρους αποτελέσματα χρόνων για κοινές τιμές, μέσω των συναρτήσεων **exists_in** και **is_almost**. Κάθε σύνολο χρόνων που ταυτίζεται σε όλα τα επιμέρους αποτελέσματα ορίζεται σαν αποτέλεσμα και αποθηκεύεται σε κατάλληλη δομή. Τέλος, το αποτέλεσμα αποθηκεύεται σε ένα αντικείμενο τύπου **pda_reply_object** και αποστέλλεται στον **pda** πράκτορα, που έκανε το ερώτημα.

- **περιεχόμενο**: **String** με τιμή **"exit"**: έχουμε εκτέλεση της συμπεριφοράς **d**, η οποία στέλνει μηνύματα τερματισμού προς τους **client/server** και στη συνέχεια τερματίζει την λειτουργία του πράκτορα **CompareAgent**.

A.2.9 Αρχείο pdaAgent.java

Ο **pdaAgent** είναι ο πράκτορας που "τρέχει" στο **pda** (ή σε άλλη κατάλληλη φορητή συσκευή). Κατά την εκκίνηση της λειτουργίας του, καλεί τη συμπεριφορά **getScenario** να εκτελεστεί, η οποία εμφανίζει στην οθόνη ένα κεντρικό μενού με τις επιλογές: **Find possible scenario** και **Exit**.
Αν ο χρήστης επιλέξει **Exit**, ο πράκτορας τερματίζει τη λειτουργία του και στέλνει μήνυμα στον πράκτορα **compareAgent**, ώστε να αναλάβει να σταματήσει τους **server/client**.
Αν ο χρήστης επιλέξει **Find possible scenario**, ο **pdaAgent** ζητά από τον πράκτορα **compareAgent** ένα πιθανό σενάριο με βάση τις τρέχουσες τιμές θερμοκρασίας στο χώρο που μας ενδιαφέρει, μέσω ενός μηνύματος τύπου **ACLMessage**. Αφού λάβει την απάντηση, διαβάζει τα περιεχόμενά του και έχουμε τις εξής περιπτώσεις:

1. Δεν υπάρχει κάποιο αποτέλεσμα: τότε εμφανίζεται ένα υπομενού με τις επιλογές **Retry**, όπου

επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to main menu**, όπου μας επιστρέφει στο κεντρικό μενού.

2. Υπάρχει ένα αποτέλεσμα: τότε εμφανίζεται το αποτέλεσμα και επιπλέον μας δίνεται η επιλογή σχετικά με το αν επιθυμούμε να παρακολουθήσουμε το σχετικό βίντεο μέσω του **Smokeview** ή όχι. Στη συνέχεια εμφανίζεται ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to main menu**, όπου μας επιστρέφει στο κεντρικό μενού.

3. Υπάρχουν περισσότερα από ένα αποτελέσματα: τότε εμφανίζεται το μήνυμα **"More than one, possible scenarios found (x)! What do you want to do?"** (όπου **x** το πλήθος των σεναρίων που βρέθηκαν) και ένα υπομενού με τις επιλογές **Show all possible scenarios**, όπου τυπώνει όλα τα πιθανά σενάρια και επιστρέφει στο κεντρικό μενού, **Exit to main menu**, όπου μας επιστρέφει στο κεντρικό μενού και **Request more values from network**, όπου ζητά από τον πράκτορα **CompareAgent** ένα πιθανό σενάριο με βάση τις τιμές θερμοκρασίας για χρονικό διάστημα **dt**. Ανάλογα με την απάντησή του αν:

   - δεν υπάρχει κάποιο αποτέλεσμα: τότε εμφανίζεται το σχετικό μήνυμα και ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to previous menu**, όπου μας επιστρέφει στο προηγούμενο υπομενού.
   - υπάρχει ένα αποτέλεσμα: τότε εμφανίζεται το αποτέλεσμα και επιπλέον μας δίνεται η επιλογή σχετικά με το αν επιθυμούμε να παρακολουθήσουμε το σχετικό βίντεο μέσω του **Smokeview** ή όχι. Στη συνέχεια εμφανίζεται ένα υπομενού με τις επιλογές **Retry**, όπου επαναλαμβάνει την προηγούμενη ενέργεια και **Exit to previous menu**, όπου μας επιστρέφει στο κεντρικό μενού.
   - υπάρχουν περισσότερα από ένα αποτελέσματα: τότε εμφανίζεται το μήνυμα **"More than one, possible scenarios found (x)! What do you want to do?"** (όπου **x** το πλήθος των σεναρίων που βρέθηκαν) και το σχετικό υπομενού όπως αυτό έχει περιγραφεί παραπάνω.

Για την επιλογή κάθε επιμέρους ενέργειας, χρησιμοποιείται η συνάρτηση **insert_choise**, η οποία δέχεται έναν χαρακτήρα από το πληκτρολόγιο.

Βιβλιογραφία

[1] J. G. Michopoulos, P. Tsompanopoulou, S. Lalis, D. Syrivelis, E. N. Houstis, A. Joshi, S. Avancha, E. N. Houstis and H. Zang *Towards an Agent Implementation of GRID-enabled and Sensor-driven Fire Dynamics Simulation.*

[2] Fire Dynamics Simulator (FDS) and Smokeview, *http://www.fire.nist.gov/fds/.*

[3] Java Agent DEvelopment Framework (JADE), *http://jade.tilab.com/.*

[4] Spyros Lalis, Dimitris Syribelis, Manos Koutsoumbelias *A simple Gateway Framework for Sensor Networks* (Computer and Communications Engineering Department, University of Thessaly).

[5] W.W. Jones, *A Review of Compartment Fire Models* (National Bureau of Standards (now NIST) Gaithersburg, Maryland, 1983).

[6] J. Quintiere, *A Perspective on Compartment Fire Growth* (Combustion Science and Technology, 1984).

[7] G.P. Forney and W.F. Moss, *Analyzing and Exploiting Numerical Characteristics of Zone Fire Models* (Fire Science and Technology, 1994).

[8] S.V. Patankar, *Numerical Heat Transfer and Fluid Flow* (Hemisphere Publishing, New York, 1980).

[9] R.G. Rehm and H.R. Baum, *The Equations of Motion for Thermally Driven, Buoyant Flows* (Journal of Research of the NBS, 1978).

[10] E.S. Oran and J.P. Boris, *Numerical Simulation of Reactive Flow* (Elsevier Science Publishing Company, New York, 1987).